Universidad
Carlos III de Madrid

PROYECTO FIN DE CARRERA

Ingeniería Técnica Telecomunicación,
especialidad Telemática

Porramática
Aplicación web para gestionar apuestas deportivas

Autor: Juan Alberto Delgado Zazo

Tutor: Francisco Valera Pintor

Director: Isaac Seoane Puyol

Leganés, abril de 2013

Título: Porramática. Aplicación web para gestionar apuestas deportivas
Autor: Juan Alberto Delgado Zazo
Tutor: Francisco Valera Pintor
Director: Isaac Seoane Puyol

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A mis padres Juan Antonio y María Dolores, por darme todo lo necesario para haber llegado hasta aquí sin pedir nada a cambio. Por su cariño y compresión.

A mi abuela Asunción y mi hermana Patricia, porque hacen que el día a día se viva de otra manera, por su carisma y su alegría.

A mis compañeros de universidad Carlos, Paola, Raquel, Ana, Marce, Marta, Bea, Álvaro, Yagüe, Sergio, Sonia, Sandra, Janine, Guille, José Luis, Jorge, Ramón. Porque esta etapa has sido de las más bonitas y todo os lo debo a vosotros. Gracias por hacer el día a día mucho más divertido.

En especial a Eva, por creer en mí y apoyarme en los momentos difíciles, porque sin su alegría y cariño esto no habría sido posible. Porque todo tiene sentido si estás a mi lado.

A mis tutores Francisco e Isaac, por ayudarme en todo lo posible y hacer que este proyecto haya salido adelante, por los ánimos y las enseñazas adquiridas.

Y a todos los que siempre han estado a mi lado y han compartido estos años de mi vida, también va dedicado a vosotros. Gracias.

Resumen

El objetivo final de este proyecto fin de carrera es el diseño y desarrollo de una aplicación web multidispositivo con requisitos de producto final, por lo que más allá de de ser un prototipo se pretende conseguir que pueda ser desplegada y puesta en funcionamiento de manera completamente operativa sin más desarrollos.

Porramática (así es como lo hemos llamado) es una herramienta para gestionar apuestas deportivas, en la cual los administradores son capaces de crear competiciones para que el resto de usuarios puedan apostar sobre dichas competiciones, con un seguimiento de dichas apuestas para conocer finalmente el número de aciertos, dando lugar a una clasificación donde se conocerá al ganador de la porra.

Se ha diseñado esta herramienta basada en el lenguaje de programación Java, y este a su vez en la tecnología web GWT (Google Web Toolkit), apoyados en una base de datos con motor MySQL y funcionando todo el conjunto en un servidor de aplicaciones web Tomcat.

De esta manera, cualquier usuario registrado en *Porramática* y mediante una conexión a Internet, es capaz de acceder a la aplicación y jugar sus apuestas en cualquier momento. La aplicación cuenta con un perfil en Twitter (@porramatica) donde se informa de la activación de una nueva jornada a jugar, y de los resultados que se suceden en la misma.

De la misma forma, si el usuario así lo prefiere, también puede recibir e-mails con las nuevas activaciones de jornadas y de los resultados de las mismas desde el remitente porramatica@gmail.com.

Abstract

The ultimate goal of this final project is the design and development of a multi-device web application with final product requirements, so that beyond being a prototype is to be achieved that can be deployed and put into operation in a fully operational without further developments.

Porramática (that's what we called it) is a tool to manage sports betting in which managers are able to create competitions for the rest of users can bet on these competitions, with tracks such bets to finally know the number hit, resulting in a classification where it will meet the winner of the competition.

This tool has been designed based on the Java programming language, and this in turn web technology GWT (Google Web Toolkit), supported by a database engine MySQL and running around the set in a Tomcat web application server .

For this reason, any registered user in Porramática and through an Internet connection, you can access the application and play your bets at any time. The application has a profile on Twitter (@porramatica) which reports on the activation of a new soccer day, and the results that occur therein.

Similarly, if the user prefers, you can also receive e-mails with new activations of days and the results of the same from the sender porramatica@gmail.com.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Motivación del proyecto

Este proyecto fin de carrera surge con el propósito de desarrollar y desplegar una aplicación final que sea capaz de gestionar apuestas deportivas a través de Internet. La aplicación a desarrollar debe ser totalmente operativa para el usuario, por lo que no contemplamos un prototipo o una versión intermedia al fin que queremos conseguir. Decidimos que debe ser una aplicación web para que sea accesible desde cualquier lugar, por lo que decidimos desarrollarla mediante la tecnología GWT.

GWT es un framework que permite al desarrollador obviar problemas concisos de la programación web, que están relacionados con la antigüedad de dichos lenguajes y su arduo mantenimiento. Gracias a este framework el desarrollo es mucho más ágil y permite un mejor mantenimiento al estar escrito en un lenguaje orientado a objetos como es Java.

La idea de realizar una aplicación de gestión de apuestas deportivas en GWT era crear una aplicación competitiva con el resto de posibles opciones que hay en la web; que fuera totalmente configurable en cuanto a términos de puntuaciones y competiciones; hacerla orientada totalmente al usuario final por lo que su manejo sea muy sencillo e intuitivo; desarrollo con software libre, con el consiguiente ahorro de costes de desarrollo; y que a su vez esté relacionado con el gran boom que hay en la red que no es otro que las redes sociales, en este caso Twitter, para la transmisión de información relacionada con las apuestas.

Todo fue concebido de tal manera que la aplicación fuera capaz de dar soporte a una comunidad de usuarios basados en roles, de tal manera que los usuarios con más responsabilidad sean los encargados de permitir al resto su inclusión en la competición, así como permitir la creación de sucesivas jornadas de juego para disputar una competición totalmente configurable conforme a las exigencias de los jugadores.

1.2 Objetivos

Tal y como hemos visto en las motivaciones del proyecto, el objetivo fundamental consiste en el desarrollo de una aplicación web para gestionar apuestas deportivas, haciendo de esta totalmente configurable, permitiendo mediante un módulo destinado para ello una gestión de usuarios basada en roles, y que dicha aplicación interactúe con la red social Twitter o a través de email para que pueda informar a los usuarios sobre posibles eventos.

En base a ese objetivo principal, se proponen los siguientes objetivos parciales:

- Estudio, modelado e implementación de una aplicación web de apuestas deportivas online, con gestión de usuarios y parámetros autoconfigurables.
- Estudio de la tecnología GWT como medio para desarrollo de la aplicación.
- Estudio, modelado e implementación de una base de datos en MySQL que albergue la funcionalidad de la aplicación.
- Estudio de desarrollo de un módulo de envío de emails por parte de la aplicación.
- Estudio de “tuiteo” de información relacionado con las competiciones a disputar por la aplicación.
- Aprender a documentar las conclusiones obtenidas sobre el desarrollo de la herramienta, aquellas limitaciones que se han encontrado y cómo ha influido en la forma final de la aplicación el día de su presentación.

1.3 Fases del desarrollo

Las fases de desarrollo de las que consta este proyecto fin de carrera son las siguientes, empezando por la más antigua en orden cronológico:

- Estructura de la aplicación, definiendo como se establecen las relaciones entre las partes (contenedor web Tomcat, GWT, base de datos).

- Estudio de las características principales que debería tener la aplicación. Entre ellas cabe destacar como se realizará la gestión de usuarios; tipos de competiciones que puede albergar la aplicación; parámetros a definir para cada tipo de competición.

- Modelado de una base de datos que implemente nuestros requisitos.

- Desarrollo de cada una de las partes de la aplicación.

- Despliegue en el entorno de desarrollo.

1.4 Medios empleados

Los medios empleados para el desarrollo de este proyecto fin de carrera son los siguientes:

- Hardware.-
  1. Equipo desarrollo.- Intel Core 2 Duo T7300 2GB Ram. Windows Vista 32 bits.
  2. Equipo producción.- Equipo Linux del Dpto. Telemática de la UC3M.

- Software.-
  1. Java.- Versión 7 JRE.
  2. GWT.- SDK 2.5.0
  3. MySQL.- Versión 5.0
  4. Tomcat.- Versión 7.0
  5. Librerías extra.- JavaMail para el envío de emails, MySQL-connector para la conexión con la base de datos (JDBC).
  6. Eclipse.- Versión Juno

1.5 Estructura de la memoria

Para facilitar la lectura de la memoria y su correcta interpretación, se incluye a continuación un breve resumen de cada capítulo:

- En el capítulo 2 hablamos del estado del arte del proyecto. Destacamos la utilización de la tecnología GWT como base de desarrollo del proyecto, escrita en el lenguaje Java, apoyándonos en una base de datos con motor MySQL, un contenedor de aplicaciones web Tomcat y una aplicación para Twitter que permite el tuiteo de información de la aplicación en la cuenta dedicada a ello.

- En el capítulo 3 trataremos extensamente el desarrollo de la aplicación. Desde como se relacionan los módulos que lo integran, su correcta configuración, hasta los aspectos que dotan de tecnología al proyecto como es el tuiteo de información

y el envío de emails. También se ilustrará la forma de utilizar la aplicación por parte del usuario.

- En el capítulo 4 se explicarán las pruebas realizadas sobre la aplicación. Pruebas locales a medida que avanzaba el desarrollo para detectar fallos de software, hasta su despliegue en el equipo de producción donde se realiza una batería de pruebas, se analiza su rendimiento y posibles fallos de ejecución. Para terminar este capítulo se explicará en detalle el desarrollo de una competición real para demostrar el correcto funcionamiento de la aplicación.

- En el capítulo 5 se detalla la historia del proyecto, tiempo empleado en su desarrollo y su presupuesto final.

- Y por último en el capítulo 6 trataremos las posibles líneas futuras de investigación y ampliación del proyecto, tanto para mejorar la versión en curso como para dotarla de nuevas funcionalidades.

Como parte de la memoria se ha incluido un apéndice para el glosario de términos que aclara toda la terminología tratada en siglas durante el desarrollo del proyecto.

Finalmente se incluyen las referencias consultadas en el desarrollo del proyecto, que incluyen las especificaciones a las que ha sido necesario recurrir para implementar aspectos concretos de las aplicaciones desarrolladas, libros y recursos web.

Capítulo 2

Estado del arte

2.1 Introducción

En este capítulo expondremos las principales tecnologías base para el desarrollo del proyecto. Como bien comentamos en el capítulo de introducción, la aplicación está desarrollada con la tecnología GWT, escrita esta a su vez en lenguaje Java, por lo que abordaremos estos dos aspectos y sus principales características.

El porqué elegimos GWT como objetivo de desarrollo del proyecto frente a otras tecnologías es debido a las múltiples ventajas que posee este framework, entre ellas por ser código abierto como comentamos en el capítulo introductorio; la compilación hace que funcione el código JavaScript en cualquier navegador, por lo que implica ser multidispositivo utilizando un navegador de Internet; depuración en tiempo real con el entorno de desarrollo elegido; y la más importante el prescindir de programar en JavasScript hace que el desarrollo sea mucho más rápido. [10]

En el apoyo de datos tenemos la base de datos MySQL, presentando sus principales características y el porqué de su elección para este proyecto. El conjunto de la aplicación, al ser una aplicación web, debe estar en un contenedor propiamente dicho, que en este caso ha sido Tomcat, por lo que también contaremos un poco su historia, características y ventajas/desventajas frente a otros similares.

Por último comentaremos los aspectos relacionados al envío de emails y tuiteo por parte de la aplicación, que librerías y aplicaciones utilizamos para ello y su historia.

2.2 Java

2.2.1 Introducción

Java es toda una tecnología orientada al desarrollo de software con el cual podemos realizar cualquier tipo de programa. Hoy en día, la tecnología Java ha cobrado mucha importancia en el ámbito de Internet gracias a su plataforma J2EE. Pero Java no se queda ahí, ya que en la industria para dispositivos móviles también hay una gran acogida para este lenguaje. La tecnología Java está compuesta básicamente por 2 elementos: el lenguaje Java y su plataforma.

Con plataforma nos referimos a la máquina virtual de Java (Java Virtual Machine). Java también es un lenguaje de programación. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía Sun Microsystems con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras. [1]

2.2.2 Historia

En 1991, la empresa Sun Microsystems crea el lenguaje Oak (de la mano del llamado proyecto Green). Mediante este lenguaje se pretendía crear un sistema de televisión interactiva. Este lenguaje sólo se llegó a utilizar de forma interna. Su propósito era crear un lenguaje independiente de la plataforma y para uso en dispositivos electrónicos. Sun deseaba un lenguaje para programar pequeños dispositivos electrónicos. La dificultad de estos dispositivos es que cambian continuamente y para que un programa funcione en el siguiente dispositivo aparecido, hay que rescribir el código. Por eso Sun quería crear un lenguaje independiente del dispositivo.

La promesa inicial de Sun era *Write Once, Run Anywhere* (escríbelo una vez, ejecútalo en cualquier lugar), proporcionando un lenguaje independiente de la plataforma y un entorno de ejecución (la JVM) ligero y gratuito para las plataformas más populares de forma que los binarios (bytecode) de las aplicaciones Java pudiesen ejecutarse en cualquier plataforma.

De esta manera se consigue paliar el problema fundamental del C++; que consiste en que al compilar se produce un fichero ejecutable cuyo código sólo vale para la plataforma en la que se realizó la compilación.

El entorno de ejecución era relativamente seguro y los principales navegadores web pronto incorporaron la posibilidad de ejecutar applets Java incrustadas en las páginas web.

En 1995 pasa a llamarse Java y se da a conocer al público. Adquiere notoriedad rápidamente. Java pasa a ser un lenguaje totalmente independiente de la plataforma y a la

vez potente y orientado a objetos. Esa filosofía y su facilidad para crear aplicaciones para redes TCP/IP ha hecho que sea uno de los lenguajes más utilizados en la actualidad. [1.]

Java ha experimentado numerosos cambios desde la versión primigenia, JDK 1.0, así como un enorme incremento en el número de clases y paquetes que componen la biblioteca estándar. [2]

Figura 2.1. Logo de Java

2.2.3 Características principales

Como hemos comentado en la historia de esta tecnología, lo que motiva a la creación de Java, en resumen, son lo siguiente:

- Creciente necesidad de interfaces mucho más cómodas e intuitivas que los sistemas de ventanas que proliferaban hasta el momento.

- Fiabilidad del código y facilidad de desarrollo. Se observó que muchas de las características que ofrecían C o C++ aumentaban de forma alarmante el gran coste de pruebas y depuración.

- Enorme diversidad de controladores electrónicos. Los dispositivos electrónicos se controlan mediante la utilización de microprocesadores de bajo precio y reducidas prestaciones, que varían cada poco tiempo y que utilizan diversos conjuntos de instrucciones. Java permite escribir un código común para todos los dispositivos.

Por ello, y con estas premisas en mente, se crea Java, aportando las siguientes características al lenguaje, que en base a ajustarnos al tamaño de la memoria mencionamos las más importantes:

- Orientado a objetos. Quizá la característica más importante junto con la independencia de plataforma. Permite fabricar programas de forma más parecida al pensamiento humano. De hecho simplifica el problema dividiéndolo en objetos y permitiendo centrarse en cada objeto, para de esa forma eliminar la complejidad.

Cada objeto se programa de forma autónoma y esa es la principal virtud. Con ello se facilita la reutilización de código y un mantenimiento óptimo.

- Independencia de la plataforma. Significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo. Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como "bytecode", que no son más que instrucciones máquina simplificadas y específicas de la plataforma Java. Esta pieza está "a medio camino" entre el código fuente y el código máquina que entiende el dispositivo destino. El bytecode es ejecutado entonces en la máquina virtual, un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código. Además, se suministran bibliotecas adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hilos, la interfaz de red, etc.) de forma unificada.

- Distribuido. Se mencionó en la historia de Java. Permite crear aplicaciones de red fácilmente. Esto es debido a que proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

- Recolector de basura. El programador determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java es el responsable de gestionar el ciclo de vida de los objetos. El programa, u otros objetos pueden tener localizado un objeto mediante una referencia a éste. Cuando no quedan referencias a un objeto, el recolector de basura de Java borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas (ejemplo: un objeto creado y únicamente usado dentro de un método sólo tiene entidad dentro de éste; al salir del método el objeto es eliminado).

- Robusto. Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y el recolector de basura mencionado elimina la necesidad de liberación explícita de memoria.

- Multihilo. Java soporta sincronización de múltiples hilos de ejecución (*multithreading*) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos. [2] [3]

2.2.4 Versiones

Prosiguiendo con la explicación del lenguaje Java, es recomendable que el usuario sepa los diferentes tipos de versiones disponibles del lenguaje que hay dependiendo del

área de desarrollo. Una versión de Java no es más que un conjunto de APIs que en su conjunto están enfocadas a un desarrollo en concreto de aplicaciones. ¿Por qué es necesario saber esto? Porque en nuestro caso que es el que nos interesa, debemos tener claro que tipo de versión debemos utilizar para poder desarrollar una aplicación como es *Porramática*. [4]

Los diferentes tipos de versiones que hay actualmente son los siguientes:

- Java Standard Edition (JSE). Es la edición que se emplea en ordenadores personales. Se le conoce también como Java Desktop (escritorio). Java Standard Edition o Java SE (conocido anteriormente hasta la versión 5.0 como Plataforma Java 2, Standard Edition o J2SE), es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java. En el desarrollo de este proyecto esta ha sido la versión utilizada, en nuestro caso la versión 7.0.

- Java Micro Edition (JME). La edición que se emplea en dispositivos móviles. Es una versión recortada del Java SE con ciertas extensiones enfocadas a las necesidades particulares de esos tipos de dispositivos. La plataforma Java Micro Edition, o Java ME (anteriormente J2ME), es una colección de APIs en Java orientadas a productos de consumo como PDAs, teléfonos móviles o electrodomésticos. Java ME se ha convertido en una buena opción para crear juegos en teléfonos móviles debido a que se puede emular en un PC durante la fase de desarrollo y luego subirlos fácilmente al teléfono. Al utilizar tecnologías Java el desarrollo de aplicaciones o videojuegos con estas APIs resulta bastante económico de portar a otros dispositivos.

- Java Enterprise Edition (JEE). Es un grupo de especificaciones (APIs) diseñadas por Sun que permiten la creación de aplicaciones empresariales, esto sería: acceso a base de datos (JDBC), utilización de directorios distribuidos (JNDI), acceso a métodos remotos (RMI), funciones de correo electrónico (JavaMail), aplicaciones Web (JSP y Servlets) etc. Aquí es importante notar que JEE es solo una especificación, esto permite que diversos productos sean diseñados alrededor de estas especificaciones como Tomcat. En *Porramática* utilizamos aspectos de JEE como es JavaMail para el envío de emails, Tomcat como contenedor web y ciertos aspectos de RMI para la serialización de objetos entre parte servidora y cliente de GWT (lo veremos en GWT).

Como hemos visto, para el desarrollo de *Porramática* hemos utilizado principalmente **Standard Edition**, aunque también hemos utilizado aspectos de Enterprise como JavaMail, el conector JDBC para la conexión con la base de datos, o el propio contenedor Tomcat donde está desplegada la aplicación. En nuestro caso, sólo hemos utilizado algunas APIs necesarias para la conexión con base de datos, envío de emails y cosas puntuales.

Hacer un uso de JEE implica necesariamente utilizar JSE pero no al revés, ya que una contiene a la otra. Esto lo podemos observar perfectamente en la siguiente imagen:

Figura 2.2. Versiones Java

Como es lógico y debido al tamaño de esta memoria, no podemos explicar en este capítulo dedicado a Java todos sus aspectos de programación, sentencias, variables y demás aspectos técnicos, por lo que en la siguiente sección vamos a explicar muy brevemente algunas APIs que hemos mencionado anteriormente y que son utilizadas en *Porramática*.

2.2.5 JavaMail

JavaMail es una API opcional a la versión estándar de Java que proporciona funcionalidades de correo electrónico, a través del paquete **javax.mail**. Permite realizar desde tareas básicas como enviar, leer y componer mensajes, hasta tareas más sofisticadas como manejar gran variedad de formatos de mensajes y datos, y definir protocolos de acceso y transporte.

JavaMail soporta, por defecto, los protocolos de envío y recepción SMTP, IMAP, POP3 y las versiones seguras de estos protocolos SMTPS, IMAPS, POP3S (estos 3 últimos a partir de la versión JDK 1.3.2), si bien puede implementar otros protocolos. El envío y recepción son independientes del protocolo, aunque podremos necesitar usar un protocolo u otro según nuestras necesidades.

Algunas de las prestaciones a destacar que ofrece este potente API son las siguientes:

- Componer y enviar un mensaje: el primer paso que ha de realizar cualquier aplicación que pretenda enviar un correo electrónico es componer el mensaje. Es posible crear un mensaje de texto plano, es decir, un mensaje sin adjuntos que contenga exclusivamente texto formado por caracteres ASCII; pero es igualmente sencillo componer mensajes más completos que contengan adjuntos. También se pueden componer mensajes que contengan código HTML e incluso imágenes embebidas en el texto.

- Descargar Mensajes: se pueden descargar los mensajes desde la carpeta que se indique ubicada en el servidor que en ese momento se esté usando. Estos mensajes pueden ser de texto plano, contener adjuntos, HTML, etc.

- Usar flags (banderines): para indicar, por ejemplo, que un mensaje ha sido leído, borrado, etc., en función de cuáles de estos flags estén soportados por el servidor.

- Establecer una conexión segura: actualmente algunos servidores de correo requieren de una conexión segura, ya sea para descargar el correo almacenado en ellos, o para enviar mensajes a través de ellos. JavaMail ofrece la posibilidad de establecer este tipo de conexiones, indicando que se trata de una conexión segura, además de poder indicar otros parámetros, en algunos casos necesarios, como el puerto donde establecer la conexión. Esta capacidad está disponible desde la versión de JDK 1.3.2.

- Autenticarse en un servidor: ciertos servidores requieren autenticación ya no sólo para leer sino también para enviar. JavaMail ofrece también la posibilidad de autenticación a la hora de enviar un correo.

- Definir protocolos: JavaMail soporta por defecto los protocolos de envío y recepción SMTP, IMAP, POP3, (y sus correspondientes seguros a partir de la versión de JDK 1.3.2), si bien puede implementar otros protocolos.

Como podemos observar posee casi todas las características deseables para realizar cualquier acción con los emails. [5]

Porramática hace uso de este API para recordar a los usuarios de la activación de una nueva jornada a jugar, así como de los resultados que se vayan sucediendo en los partidos conforme el administrador vaya actualizándolos. Esto es totalmente opcional a los usuarios, ya que estos al ser creados se puede indicar mediante un checkbox si permiten o no la recepción de emails por parte de la aplicación.

Para este uso, hemos creado una cuenta de correo en GMail llamada porramatica@gmail.com que será la encargada de enviar los emails con este remitente.

En los siguientes subapartados detallaremos brevemente como instalar este API, su funcionamiento y como configurarlo para el envío de emails.

2.2.5.1 Instalación

Para la utilización de este API, debemos descargarnos el zip de la página oficial de JavaMail e incluirlo en el directorio de librerías de la aplicación (se detalla en el capítulo 3 la estructura de librerías de la aplicación). El archivo que nos interesa del zip es el llamado mail.jar, que contiene los métodos y funcionalidades necesarias para nuestro uso, incluidos los soportes para los protocolos IMAP, POP3 y SMTP así como las versiones seguras. [6]

Obviamente debemos tener instalada una versión de JSE para la utilización de este API, o en su defecto una JEE tal y como comentamos en las versiones de Java y como se relacionan unas con otras. En este caso debemos tener una versión superior a la 1.1 para que JavaMail funcione en JSE, y en el caso de JEE a partir de la 1.3 ya viene incorporada, por lo que comentamos del uso de esta API en JEE. Para el caso de *Porramática* no tenemos ningún problema puesto que usamos la versión 7.0.

2.2.5.2 Envío de un email

Para el envío de emails, que es la funcionalidad escogida para este proyecto, se ha de proceder de la siguiente forma:

- Crear un objeto Session y mediante su constructor pasarle los parámetros de conexión, entre ellos los del servidor SMTP.

- Autenticarse si es necesario, en nuestro caso si lo es al tratarse de una cuenta de correo de GMail y utilizar conexión segura.

- A continuación se crea el mensaje y se rellenan sus campos (asunto, emisor, receptores, y cuerpo del mensaje).

- Por último se envía el mensaje utilizando el objeto Transport.

La configuración que específica Google para el servicio de GMail son las que se describen en la siguiente figura:

Incoming Mail (POP3) Server - requires SSL:	pop.gmail.com **Use SSL**: Yes **Port**: 995
Outgoing Mail (SMTP) Server - requires TLS or SSL:	smtp.gmail.com **Use Authentication**: Yes **Port for TLS/STARTTLS**: 587 **Port for SSL**: 465
Server timeouts	Greater than 1 minute, we recommend 5
Full Name or Display Name:	[your name]
Account Name or User Name:	your full email address (including @gmail.com or @your_domain.com)
Email Address:	your email address (username@gmail.com or username@your_domain.com)
Password:	your Gmail password

Figura 2.3. Configuración cliente email

Al objeto Properties habrá que configurarle que **se requiere autenticación** (usuario y contraseña de la cuenta porramatica@gmail.com). Se establece **TLS activado**, por consiguiente el **puerto** a asignar es el **587**. Sólo nos queda establecer la propiedad del **servidor SMTP**, que para GMail es smtp**.gmail.com**.

Una vez establecidas las propiedades, se instancia un objeto Session al que se le pasan estas propiedades junto con el usuario y contraseña de la cuenta para que autentique.

El siguiente paso es crear el mensaje. Se asigna un asunto, un cuerpo, la dirección desde la cual se manda el email (porramatica@gmail.com) y los destinatarios, que hemos decidido que sean con copia oculta para que el usuario únicamente vea el remitente del email.

Sólo nos queda instanciar al objeto Transport y mandar el email. [6] [7]

2.2.6 JDBC

Una de las principales aplicaciones de cualquier lenguaje moderno es la posibilidad de utilizar datos pertenecientes a un sistema de base de datos. La dificultad del manejo de archivos y las facilidades de manejo de datos que ofrecen los sistemas gestores de base de datos (SGBDs) son los causantes de esta necesidad.

En el mercado hay gran cantidad de bases de datos y cada una de ellas se maneja de un modo diferente. Esto está en contra del planteamiento fundamental de Java que intenta que la programación sea independiente de la plataforma.

Hoy en día hay que tener en cuenta que la inmensa mayoría de los SGBD administran **bases de datos relacionales**. Éstas son bases de datos que permiten organizar los datos en tablas que después se relacionan mediante campos clave y que trabajan con el lenguaje estándar conocido como SQL.

Porramática hace uso de una base de datos relacional muy compleja, ya que conlleva desde la gestión de usuarios y sus roles, para posteriormente especificar cada competición, jornada, cada una de ellas bien parametrizada, así como todas las apuestas que cada usuario refleja sobre ella.

Pero centrándonos de nuevo en el tema que nos trata, JDBC, al igual que pasaba con JavaMail, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java. El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos, en este caso para este proyecto es una base de datos con motor MySQL. [8]

En la siguiente figura se puede apreciar este aspecto:

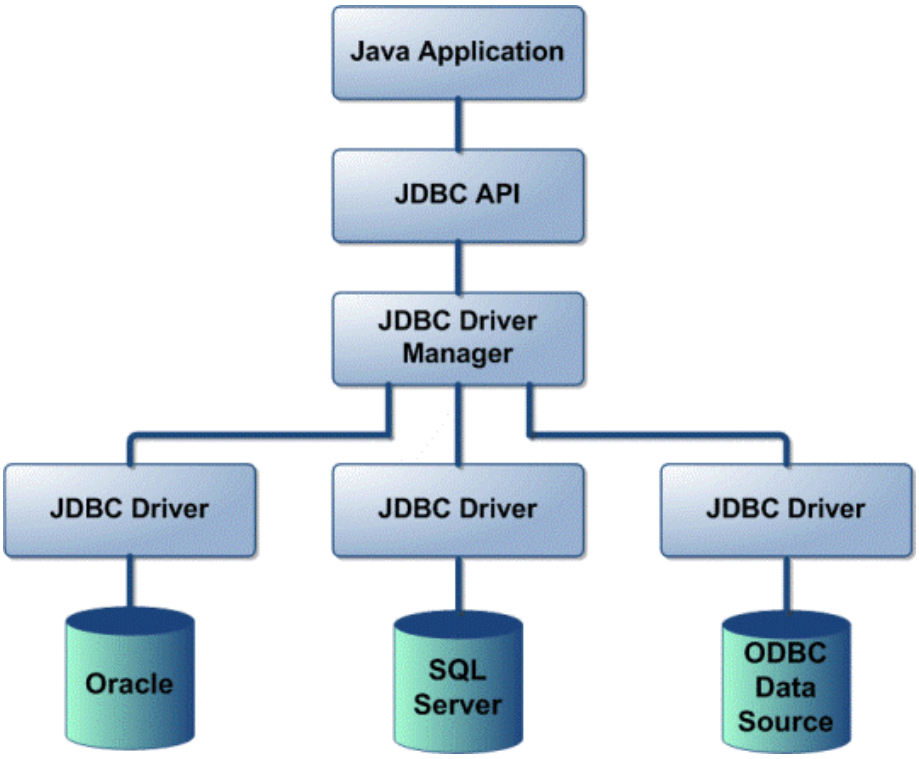

Figura 2.4. Comunicación JDBC con bases de datos

2.2.6.1 Instalación

Para poder instalar el conector JDBC primero debemos descargarnos su versión correspondiente dependiendo de la base de datos a usar, en este caso MySQL. [9]

Una vez descargado, descomprimiremos el zip en el directorio de librerías de la aplicación (se detalla en el capítulo 3 la estructura de librerías de la aplicación). Si estamos usando un entorno de desarrollo, hay que incluirlo en el buildpath de nuestra aplicación para que esta sepa donde encontrar las clases necesarias.

2.2.6.2 Funcionamiento

En *Porramática* disponemos de un servicio únicamente destinado a la comunicación con la base de datos. Es decir, esta clase es la encargada de realizar todas las tareas de consultas, inserciones y verificaciones a petición por la aplicación.

Para conseguir conectar la base de datos con nuestra aplicación, esta requiere el URL de la base de datos y las propiedades que establezca nuestro controlador JDBC. Las clases necesarias para usar JDBC están en el paquete java.sql.

El primer paso es instalar el controlador (*driver*) de la base de datos. Para ello se lanza el controlador en la propia aplicación mediante el método estático forName de la clase Class. Al ser un motor MySQL el método forName recibe por parámetro el String “com.mysql.jdbc.Driver”.

Una vez instalado el controlador, debemos establecer conexión con nuestra base de datos. Al configurar por primera vez el motor MySQL (lo veremos en la subsección

dedicada a ello) nos pedirá un nombre de usuario y contraseña para dicha instancia del motor. Por lo tanto la URL quedaría de la siguiente forma:

*jdbc:mysql://**host**:**port**/**name_db**?user=**xx**&password=**yy***

- En nuestro caso, al desplegar *Porramática* en local, podemos poner en host "localhost".
- El puerto por defecto de MySQL es 3306.
- El nombre de la base de datos es el que hayamos elegido para ello.
- Nombre de usuario
- Y por último contraseña establecida.

Esta URL es la que recibe el DriverManager para realizar la conexión, y si todo funciona correctamente y no hay excepciones, nos devolverá un objeto Connection con el que podremos crear un objeto Statment para empezar a operar sobre la base de datos a través de ResultSet.

Veamos un ejemplo de cómo quedaría la conexión con la base de datos utilizando el API de JDBC:

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/usuarios?user=pepe&&password=pepe1234");
Statement st=con.createStatement();
```

Como el objetivo de esta subsección no es explicar en detalle todo el API, tán solo mencionar muy brevemente que a través de los objetos ResultSet y el objeto Statement creado después de realizar la conexión, podremos empezar a iterar sobre la base de datos con consultas mediante el método *executeQuery*, o con inserciones, actualizaciones o eliminaciones mediante *executeUpdate*.

2.3 GWT (Google Web Toolkit)

2.3.1 Introducción

Google Web Toolkit es un entorno de desarrollo Java, mas concretamente un framework (estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado) basado en software libre y que permite escribir aplicaciones AJAX fácilmente.

GWT permite escribir las aplicaciones en el lenguaje de programación Java, y luego se encarga de compilarlo, traduciendo la parte del cliente a lenguaje de programación JavaScript + HTML + CSS, generando código JavaScript más eficiente que el escrito a mano.

El ciclo de desarrollo, resumido brevemente, de una aplicación GWT sería el siguiente:

- Desarrollar la aplicación en Java usando cualquier entorno de desarrollo y las bibliotecas de GWT. Se puede probar y depurar sobre cualquier JVM.

- Usar las herramientas que contiene GWT para convertir el código a Javascript y HTML. La herramienta principal es el compilador de GWT que veremos más adelante.

- Probar la aplicación en varios navegadores web. El código generado se puede usar en cualquier tipo de navegador sin restricciones.

- Desplegarla en un servidor HTTP. Al ser una aplicación web, esta es capaz de ser desplegada en cualquier contenedor web, en el caso de *Porramática* en Tomcat.

En las siguientes secciones veremos las características principales de esta tecnología, como se comunican sus partes y así entenderemos la estructura de *Porramática*. [10]

2.3.2 Historia

JavaOne es una conferencia anual (desde 1996) puesta por Sun Microsystems para discutir las tecnologías de Java, sobre todo entre los desarrolladores de Java. Google anunció la biblioteca GWT en la conferencia JavaOne de 2006 y lanzó la versión 1.0 RC 1 (build 1.0.20) el 16 de mayo de 2006.

La última versión disponible actualmente es la 2.5.1 que fue lanzada en Enero de 2013.

2.3.3 Características principales

Vamos a mencionar algunas de las características más reseñables de GWT:

- RPC realmente fácil. Para comunicarse desde el navegador que lanza la aplicación con el servidor web, solamente se necesita definir clases de Java serializables para las peticiones y respuestas. En producción, GWT serializa automáticamente las peticiones del navegador y des-serializa las repuestas desde el servidor web. El mecanismo de RPC de GWT puede incluso manejar jerarquía de polimorfismo en clases, y puedes manejar las posibles excepciones.

- Componentes de la interfaz de usuario dinámico y re-utilizables. Crea un widget para construir otros. Coloca los widgets automáticamente en paneles. Envía tus widget a otros desarrolladores en archivos JAR.

- Administración del historial del navegador. Las aplicaciones en AJAX no necesitan utilizar el botón “atrás” (back) del navegador. Y GWT no es la excepción, es decir, no es necesario llamar a otras páginas para realizar las diferentes acciones, ni recargar el navegador.

- Depuración en tiempo real. Para cuando la aplicación esté lista, el código de la misma es traducido a JavaScript, pero mientras se está desarrollando este corre sobre una JVM. Lo que significa que en la fase de desarrollo se tiene la posibilidad de depurar la aplicación con los avanzados sistemas de debugging y manipulación de excepciones incluidos en entornos de desarrollo como Eclipse.

- Compatibilidad con los navegadores. Las aplicaciones en GWT serán automáticamente soportadas por navegadores como Firefox, Internet Explorer, Mozilla, Safari, y Opera sin ningún tipo de operación para la detección de los mismos, en la mayoría de los casos.

- Integración con Junit. Mediante la integración de JUnit en GWT se pueden probar las aplicaciones y depurarlas en un navegador mientras se construyen, incluso, se puede testear llamadas asíncronas a procedimientos remotos RPC.

- Internacionalización. Crea aplicaciones y librerías de internacionalización rápida y fácilmente.

- Interoperability and fine-grained control. Si las librerías de clases de GWT no son suficientes, se puede mezclar JavaScript en el código de la aplicación usando la interfaz nativa de scripts de Java (JavaScript Native Interface, JSNI).

- GWT es un proyecto de código abierto. Todo el código de GWT está disponible bajo la licencia Apache 2.0.

2.3.4 Arquitectura

Es importante para el lector que no conoce GWT que tenga en mente la arquitectura de esta, para poder así esclarecer con más idea como se estructura *Porramática*.

En la siguiente figura podemos ver como es la arquitectura:

Figura 2.5. Arquitectura GWT

Describimos a continuación cada una componente:

- Compilador GWT Java-a-JavaScript. El compilador GWT Java-a-JavaScript traduce del lenguaje de programación Java a JavaScript. El compilador se utiliza cuando necesitamos desplegar la aplicación en modo web.

- Navegador web “Hosted” de GWT. El Navegador web “Hosted” de GWT permite correr y ejecutar aplicaciones GWT en modo hosted, donde lo que estás corriendo son bytecodes de Java sobre una máquina virtual sin compilarlos a JavaScript.

- Emulación de librerías JRE. GWT contiene implementaciones en JavaScript de las librerías de clases más usadas en Java, incluyendo la mayoría de las clases del paquete java.lang y un subconjunto de clases del paquete java.util. El resto del estándar de librerías de Java no es soportado nativamente con GWT. Por ejemplo, las clases de los paquetes como java.io no se utilizan en aplicaciones web ya que estas acceden a recursos en la red y al sistema de archivos local.

- Librería de clases de interfaz de usuario de GWT. Las librerías de clases de interfaz de usuario de GWT son un conjunto de interfaces y clases personalizadas que permiten crear "widgets" para el navegador, como botones, cajas de texto, imágenes, y texto. Éste es el núcleo de las librerías de interfaz de usuario para crear aplicaciones GWT.

Una vez que hemos visto la arquitectura, vamos a ver en las siguientes subsecciones tres aspectos fundamentales para terminar de entender el funcionamiento de GWT, que son el compilador, encargado de la traducción de Java a JavaScript; diferenciar entre el lado del cliente y el servidor; y posteriormente unido a esto como se realiza la comunicación entre estos dos últimos.

2.3.5 Compilador

El corazón de Google Web Tooltik es un compilador que traduce el código Java en JavaScript, convirtiendo así la aplicación desarrollada en Java en una aplicación equivalente en JavaScript. En pocas palabras, si la aplicación Google Web Tooltik compila y ejecuta en "hosted mode" como se esperaba, y si Google Web Tooltik compila/traduce la aplicación a JavaScript sin problemas entonces la aplicación está lista para correr sobre un navegador web desde cualquier servidor http.

El compilador Google Web Tooltik soporta la mayoría de las características del lenguaje de programación Java. Mientras que las librerías Google Web Tooltik runtime emulan una gran parte de las librerías de Java. [11]

Vamos a ver algunas consideraciones a tener en cuenta al desarrollar en GWT y que difieren de la forma en que se hace normalmente desarrollando una aplicación Java normal:

- Tipos de datos nativos. Byte, char, short, int, long, float, double, Object y String son soportados. Sin embargo, no existen tipos integrales de 64-bit integral en JavaScript, así que las variables de tipo long son mapeadas a variables de coma flotante en JavaScript. Para asegurar la máxima consistencia entre el “modo hosted” y el “modo web”, se recomienda que el uso de variables tipo int.

- Excepciones. Try, catch, finally y excepciones definidas por el usuario son soportadas normalmente, aunque Throwable.getStackTrace () no es soportada por el modo web.

- Multi-hilado (subprocesamiento múltiple) y sincronización. Los intérpretes de JavaScript usan single-threaded (uso de hilos simple), así que cuando Google Web Toolkit acepta la palabra synchronized, en realidad esto no tiene ningún efecto sobre la aplicación final. Los métodos de las librerías de sincronización no están disponibles, incluyendo Object.wait (), Object.notify (), y Object.notifyAll ().

Como acabamos de ver hay que tener cuidado a la hora de programar en GWT, ya que al estar programando sobre Java no nos damos cuenta de los pequeños detalles que no soporta GWT.

Aún así la potencia del compilador y como este es capaz de ahorrarnos tanto tiempo a la hora de transcribir el código supera con creces la atención que debemos poner sobre estos detalles.

2.3.6 Parte cliente y parte servidor

Hemos dado un repaso importante a GWT, sus características principales y su arquitectura, así que nos queda en profundizar en como se ejecuta ese código en nuestra máquina desplegada, y como se comunican las partes que lo integran.

Una vez terminada, la aplicación será enviada por la red a un usuario, donde correrá como JavaScript dentro del navegador de este, tarea que realizó el compilador. Todo lo que suceda dentro del navegador de un usuario es llamado procesamiento del lado del cliente, o clientside processing.

Es importante recordar que en el lado del cliente se usa JavaScript, por lo que hay que usar solamente librerías y lenguaje Java que puedan ser traducidos, teniendo en cuenta las consideraciones vistas en la subsección anterior. [11] [12]

La serialización es un paso muy importante, ya que se encarga de convertir un objeto en su representación textual de tal manera que esta representación mantenga el estado en el cual se encontraba el objeto al momento de ser serializado. Una vez serializado la cadena de texto resultante se debe poder convertir de regreso al objeto programático del cual fue creado y en el estado en el cual este se encontraba al momento de ser serializado. Las clases de los objetos a serializar deben implementar de IsSerializable de GWT para la parte del cliente, o de Serializable de java.io si es una clase del servidor.

En la parte opuesta tenemos el lado del servidor, también llamado server-side processing. Cuando la aplicación necesita interactuar con el servidor (por ejemplo, para cargar o guardar datos, en nuestra caso interaccionar con la base de datos), esta realiza una petición del lado del cliente (client-side request) desde el navegador, a través de la red usando invocaciones remotas a métodos (remote procedure call, RPC, que veremos en la siguiente subsección). Mientras se está procesando una llamada RPC, el servidor está ejecutando código del lado del servidor. Aquí no tenemos ninguna restricción de ningún tipo, podemos usar libremente Java en su totalidad, así como cualquier librería y API que necesitemos implementar.

2.3.7 Invocación procedimientos remotos

El concepto de RPC es muy similar al RMI. Una diferencia fundamental entre Google Web Toolkit y las tradicionales aplicaciones web, es que las aplicaciones Google Web Toolkit no necesitan de otras páginas web mientras son ejecutadas. Ya que las páginas construidas con Google Web Toolkit corren como aplicaciones sobre el navegador, éstas no necesitan hacer nuevas peticiones al servidor para, por ejemplo, realizar actualizaciones en la interfaz de usuario. Sin embargo, como todas las aplicaciones cliente/servidor, los programas en Google Web Toolkit necesitarán pedir ciertos datos al servidor para realizar determinadas tareas.

El mecanismo para interactuar con el servidor a través de la red es llamado “remote procedure call” (RPC), que viene siendo en español “llamada a procedimiento remoto”. El RPC en Google Web Toolkit permite fácilmente al cliente enviar y recibir objetos de Java sobre HTTP. Cuando es usado adecuadamente, RPC te da la oportunidad de mover

toda la lógica de la interfaz de usuario al cliente, lo que mejora el funcionamiento de la aplicación, reduce el ancho de banda usado, reduce la carga al servidor, y le presenta al usuario final una experiencia más agradable navegando por la página, ya que la carga y los recursos los mueve el servidor que es mucho más eficiente. El código del lado del servidor que es invocado desde el cliente es frecuentemente llamado “servicio”, por lo que el “llamar procedimientos remotos” es comúnmente llamado como invocación de servicios. [13]

En la siguiente figura podemos ver el funcionamiento de implementación de un servicio por parte de una aplicación:

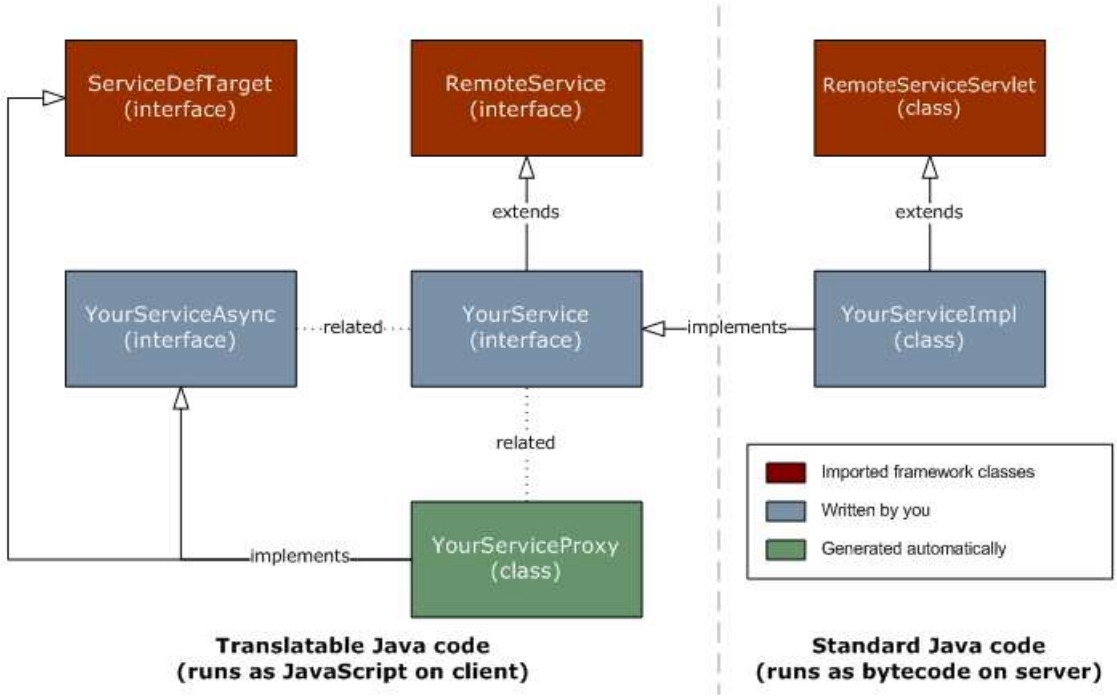

Figura 2.6. Implementación RPC

2.4 MySQL

2.4.1 Introducción

MySQL es un sistema de gestión de bases de datos relacional, multiusuario y multihilo.

En esta sección vamos a tratar resumidamente un poco su historia, como surgió y porqué, para centrarnos en sus características.

Más adelante, cuando empecemos a detallar *Porramática*, entraremos en más detalle en el diseño de la base de datos para esta aplicación, ya que fue un punto de inflexión importante en el desarrollo de la misma.

2.4.2 Historia

La empresa MySQL AB (originalmente TCX DataKonsultAB) nace en 1995, en Suecia, fundada por David Axmark, Allan Larsson, y Michael "Monty" Widenius. Monty llevaba varios años desarrollando un sistema había que ofrecía una forma optimizada y flexible para acceder a bases de datos SQL utilizando el método ISAM, ya que ninguna de las interfaces existentes le resultaba adecuada, así surgió una nueva API de acceso a SQL que podía y puede ser accedida y modificada por terceras partes, y esta API se llamó MySQL.

MySQL es un software de código abierto, licenciado bajo la GPL de la GNU, aunque MySQL AB distribuye una versión comercial, en lo único que se diferencia de la versión libre es en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de otra manera, se vulneraría la licencia GPL. [14]

2.4.3 Características principales

Inicialmente, MySQL carecía de algunos elementos esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de esto, atrajo a los desarrolladores de páginas web con contenido dinámico, debido a su simplicidad, de tal manera que los elementos faltantes fueron complementados por la vía de las aplicaciones que la utilizan. Poco a poco estos elementos faltantes, están siendo incorporados tanto por desarrolladores internos, como por desarrolladores de software libre.

Se pueden destacar las siguientes características principales:

- El principal objetivo de MySQL es velocidad y robustez.

- Soporta gran cantidad de tipos de datos para las columnas.
- Gran portabilidad entre sistemas, puede trabajar en distintas plataformas y sistemas operativos.
- Cada base de datos cuenta con 3 archivos: Uno de estructura, uno de datos y uno de índice y soporta hasta 32 índices por tabla.
- Aprovecha la potencia de sistemas multiproceso, gracias a su implementación multihilo.
- Flexible sistema de contraseñas y gestión de usuarios, con un muy buen nivel de seguridad en los datos.
- El servidor soporta mensajes de error en distintas lenguas.

Y ahora comparamos sus ventajas con sus desventajas:

Ventajas:

- Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.
- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Facilidad de configuración e instalación.
- Soporta gran variedad de sistemas operativos.
- Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.
- Conectividad y seguridad.

Desventajas:

- Un gran porcentaje de las utilidades de MySQL no están documentadas.
- No es intuitivo, como otros programas orientados al usuario como Access.

Para terminar esta subsección vamos a mencionar como hemos utilizado MySQL en *Porramática* tanto para crear la base de datos, como para exportarla en un script e importarla en el equipo de desarrollo.

2.4.4 Arranque del servicio MySQL

Arrancar el servicio MySQL es muy sencillo. Tan solo debemos abrir una consola en el equipo de producción, ir hasta el directorio /etc/init.d/ y allí, al estar la instancia de MySQL ejecutar:

mysql start –p "password"

En caso de duda, siempre se puede consultar la ayuda con *mysql –help*

2.4.5 SQLYog e Intérprete de comandos

Cuando empezamos a desarrollar el proyecto y llegamos a la parte de diseñar e implementar la base de datos, nos preguntamos que herramienta usaríamos para ello. Al ser el **equipo de desarrollo** un equipo Windows, escogimos la herramienta SQLYog. Esta herramienta GUI es muy potente y ágil de usar.

Esta opción de SQLYog es mucho más intuitiva y fácil de usar que programar directamente sobre el intérprete de comandos mysql que dispone MySQL. Vamos a explicar un poco como nos puede ayudar SQLYog en la tarea de diseño e implementación: [15]

- Editor con resaltado de sintaxis y diversas opciones de formato automático.
- Ricos menús contextuales para realizar tareas comunes sin tener que escribir SQL.
- Constructor de queries visual mediante un editor de texto.
- Juego de caracteres completo / soporte Unicode.
- Búsqueda de claves foráneas.
- Exportación de base de datos en script, con o sin datos.

Cuando necesitamos realizar una copia actual de nuestra base de datos de su estructura y sus datos, sólo tenemos que seleccionar la opción "Backup Database as SQL Dump…", para a continuación poner un nombre al archivo .sql que generará la aplicación.

En cambio, en el **equipo de producción**, al ser un equipo Unix, no tenemos esta ventaja y tenemos que recurrir al Intérprete de comandos de MySQL.

mysql es un simple shell SQL (con capacidades GNU readline). Soporta uso interactivo y no interactivo. Cuando se usa interactivamente, los resultados de las consultas se muestran en formato de tabla ASCII. Cuando se usa no interactivamente (por ejemplo, como filtro), el resultado se presenta en formato separado por tabuladores. El formato de salida puede cambiarse usando opciones de línea de comandos. [14]

Para que aparezca el prompt de mysql debemos ejecutar la siguiente instrucción en la consola del equipo:

mysql –u “user” –p “password”

Una vez que aparezca el prompt mysql, podemos hacer la carga del script que generamos con SQLYog de la siguiente forma:

source nombre_archivo.sql

2.5 Tomcat

2.5.1 Introducción

Para terminar de entender la estructura de Porramática, nos falta por explicar el servidor web Tomcat.

Tomcat, o Apache Tomcat, es un servidor web multiplataforma que funciona como contenedor de servlets y que se desarrolla bajo el proyecto denominado Jackarta perteneciente a la Apache Software Foundation bajo la licencia Apache 2.0 y que implementa las especificaciones de los servlets y de JavaServer Pages o JSP de Sun Microsystem. Dicho servidor es mantenido y desarrollado por miembros de la fundación y voluntarios independientes, los cuales tienen libre acceso al código fuente bajo los términos establecidos por la Apache Software Foundation.

Vamos a ver un poco la historia de Tomcat, sus características principales, como se organiza la aplicación, y por último como desplegamos nuestra aplicación en él.

2.5.2 Historia

Apache Tomcat comenzó siendo una implementación de servlets iniciada por James Duncan Davidson, que trabajaba como arquitecto software en Sun Microsystem y que posteriormente ayudó al proyecto de código abierto. Duncan, inicialmente pensó que el proyecto se convertiría en software de código abierto y además quiso ponerle un nombre de animal, en este caso Tomcat (gato) ya que, de algún modo pretendía trasladar la posibilidad de cuidarse por sí mismo, es decir, de ser independiente.

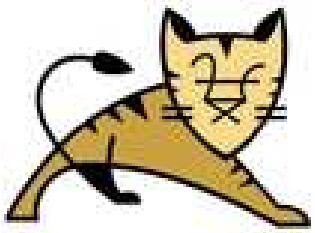

Figura 2.7. Logo de Tomcat

Las primeras distribuciones de Apache Tomcat fueron las 3.0.x aunque las versiones estables más recientes son las 6.0.36 y la 7.0.39.

2.5.3 Características principales

Como se ha dicho anteriormente, Apache Tomcat es un servidor web que da soporte a servlets y JSPs de modo que no es un servidor de aplicaciones propiamente dicho. Dado que dicho producto fue desarrollado en Java, éste puede ejecutarse sobre cualquier sistema operativo, previa instalación de la máquina virtual de Java, aunque también se puede usar con MAMPP (Mac OS X), LAMPP (GNU/Linux), WAMPP (Windows) o XAMPP (cualquier sistema operativo).

Además, puede funcionar como servidor web por sí mismo. Sin embargo en sus inicios se pensaba que dicho servidor era recomendable usarse en entornos de desarrollo con requisitos mínimos de velocidad. [16]

En la actualidad no existe esta percepción y por esto, es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Más en concreto, las principales características que soporta la última versión estable son:

- Autenticación de acceso básico.
- Negociación de credenciales.
- HTTPS
- Alojamiento compartido.
- CGI o interfaz de entrada común.
- Servlets de Java.
- SSI
- Consola de administrador.

2.5.4 Estructura

La estructura de directorios de Tomcat es la siguiente:

- bin: Arranque, parada, scripts y ejecutables.
- common: clases comunes que puede utilizar Catalina (contenedor de servlets) y las aplicaciones web.
- conf: Ficheros XML y la correspondiente DTD para la configuración de Apache Tomcat.
- logs: Logs del contenedor de servlets y de las aplicaciones.

- server: Clases usadas por el contenedor de servlets.
- shared: Clases compartidas por todas las aplicaciones web.
- webapps: Directorio que contiene las aplicaciones web.
- work: Almacenamiento temporal de ficheros y directorios.

Destacamos 3 directorios: webapps porque en este directorio es donde tenemos que copiar nuestra aplicación en formato .war para que Tomcat la despliegue; logs porque en caso de errores de ejecución es donde aparecerán los ficheros con dichos errores; y conf porque contiene ficheros de configuración en formato .xml, entre ellos el fichero para asignar roles a usuarios tomcat, y con ello a la aplicación Manager que explicamos a continuación.

2.5.5 Manager

Esta aplicación web, que viene incorporada en la instalación de Tomcat, nos va a permitir desplegar y replegar nuestras aplicaciones sin tener que entrometernos en el directorio webapps.

La ruta por defecto es:

http://host:8080/manager/html

Una vez abierto, nos pedirá que introduzcamos el usuario y contraseña de usuario de Tomcat, que ya tuviéramos que tener configurado en el fichero de configuración de usuarios del directorio conf, comentado en la subsección anterior.

Una vez dentro se nos mostrará todas las aplicaciones desplegadas en el servidor, entre ellas el manager, que no es más que otra aplicación.

Para desplegar Porramática debemos ver la siguiente figura:

Figura 2.8. Manager de Tomcat

Tan solo debemos pulsar sobre el botón “Desplegar”, seleccionamos nuestra aplicación comprimida en un .war, y a continuación pulsamos sobre “Seleccionar archivo”.

Al cabo de un tiempo nuestra aplicación estará desplegada, aparecerá en la lista de aplicaciones que maneja Tomcat y podremos ejecutarla sólo con pinchar en ella.

2.6 Twitter

2.6.1 Introducción

Twitter es una red social basada en el microblogging, con sede en San Francisco (California). Creada por Jack Dorsey en marzo de 2006 y lanzada en julio del mismo año, la red ha ganado popularidad mundialmente y se estima que tiene más de 200 millones de usuarios, generando 65 millones de tweets (publicaciones o actualizaciones del estado de un usuario mediante mensajes de texto plano con un máximo de 140 caracteres) al día y maneja más de 800.000 peticiones de búsqueda diarias [17].

Figura 2.9. Logo de Twitter

Los usuarios pueden subscribirse a los tweets de otros usuarios (a esto se le llama seguir y a los suscriptores se les llama seguidores). Por defecto los mensajes son públicos, pudiendo difundirse privadamente mostrándolos únicamente a seguidores. Los usuarios pueden twittear desde la web del servicio, desde aplicaciones oficiales externas, aplicaciones en dispositivos móviles (como los smartphones) o mediante SMS, disponible en ciertos países.

2.6.2 Historia

Twitter comenzó como un proyecto de I+D dentro de Obvious, LLC. Una pequeña start-up de San Francisco durante marzo de 2006. El nombre original del producto era twttr, inspirado por Flickr. Al principio fue usado internamente por la compañía desarrolladora hasta que lo lanzó oficialmente al público en octubre del mismo año. El servicio rápidamente ganó adeptos, y en marzo de 2007 ganó el premio South by Southwest Web Award en la categoría de blog.

Jack Dorsey es el creador de esta aplicación web y actual presidente del consejo de administración de Twitter Inc., empresa que surgió a raíz de Obvious, LLC y el éxito cosechado por Twitter. A principios de 2008, el equipo de Twitter estaba compuesto por 18 personas, durante 2009 su plantilla se multiplicó por cuatro y sigue creciendo hasta los 450 que tiene en 2011.

2.6.3 Tecnología

La interfaz web de Twitter está escrita en Ruby on Rails, y los mensajes se mantienen en un servidor que funciona con software programado en Scala y además dispone de una API abierta para todo tipo de desarrolladores. Según Twitter, más del 50% del tráfico que reciben llega a través del API. [17]

Actualmente han liberado la plataforma *Bootsrap*, que define una maquetación basada en CSS y Javascript para la creación de webs y aplicaciones y permitir de esta manera ahorrar tiempo a la hora de diseñar la web y centrarnos en la programación.

2.6.4 Mensajes

Los usuarios pueden agrupar mensajes sobre un mismo tema mediante el uso de *hashtags* (palabras o frases iniciadas mediante el uso de una "#" almohadilla). De forma similar, la "@" arroba seguida de un nombre de usuario se usa para mencionar o contestar a otros usuarios. Para volver a postear un mensaje de otro usuario y compartirlos con los propios seguidores, la función de *retweet* se marca con un "RT" en el mensaje.

A finales de 2009 se añadió la opción de listas, haciendo posible el seguir (así como el mencionar y contestar) listas de usuarios en vez de usuarios individuales.

Los mensajes fueron fijados a 140 caracteres máximo para añadir compatibilidad con los mensajes SMS, ya que a través de este sistema también es posible actualizar el estado.

Capítulo 3

Desarrollo de la aplicación *Porramática*

3.1 Introducción

En este capítulo describimos detalladamente el desarrollo de la aplicación, paso a paso, desde el concepto inicial y su arquitectura, pasando por el diseño e implementación de la base de datos, así como librerías empleadas y APIs que vimos en el estado del arte tanto para el envío de emails y tuiteo como el acceso a base de datos.

La aplicación, desarrollada en GWT, tiene una parte *frontend* (lado cliente) que interactúa con el usuario y realiza peticiones al lado servidor; y una parte *backend* (lado servidor) que recibe peticiones del lado cliente y las contesta, para realizar acciones sobre la base de datos o para el envío de emails y tuits.

Como comentamos en el capítulo introductorio, la aplicación va a ser capaz de gestionar apuestas deportivas, permitiendo que los administradores puedan dar de alta competiciones con jornadas sobre las que apostar, y los usuarios serán los encargados de apostar sobre dichas jornadas, para posteriormente visualizar las apuestas realizadas individualmente o en una clasificación con los demás apostantes. Más adelante se explicará con más detalle, en la sección de módulos funcionales, los tipos de competiciones que soporta la aplicación, tipos de jornadas a jugar, donde estudiaremos en detalle cada módulo funcional de la aplicación y sus múltiples configuraciones.

3.2 Arquitectura

3.2.1 Arquitectura tecnológica

La arquitectura tecnológica está basada en un servidor web Tomcat, con versión 7.0, donde corre toda la aplicación, apoyado en una base de datos MySQL, con versión 5.0.

El usuario puede acceder como cliente desde cualquier plataforma, es decir un navegador web, ya sea un PC de sobremesa, un portátil, un tablet o un smartphone, ya que la interfaz de la aplicación es HTML por estar desarrollado en GWT, lo que la hace accesible desde cualquier dispositivo que tenga un navegador que soporte el estándar.

Como ya mencionamos en la arquitectura de GWT, el usuario ejecutará su lado cliente en el navegador, y realizará peticiones al lado servidor mediante llamadas a procedimiento remoto, y esta última interacionará con la base de datos.

En la siguiente figura podemos ver lo descrito de manera gráfica:

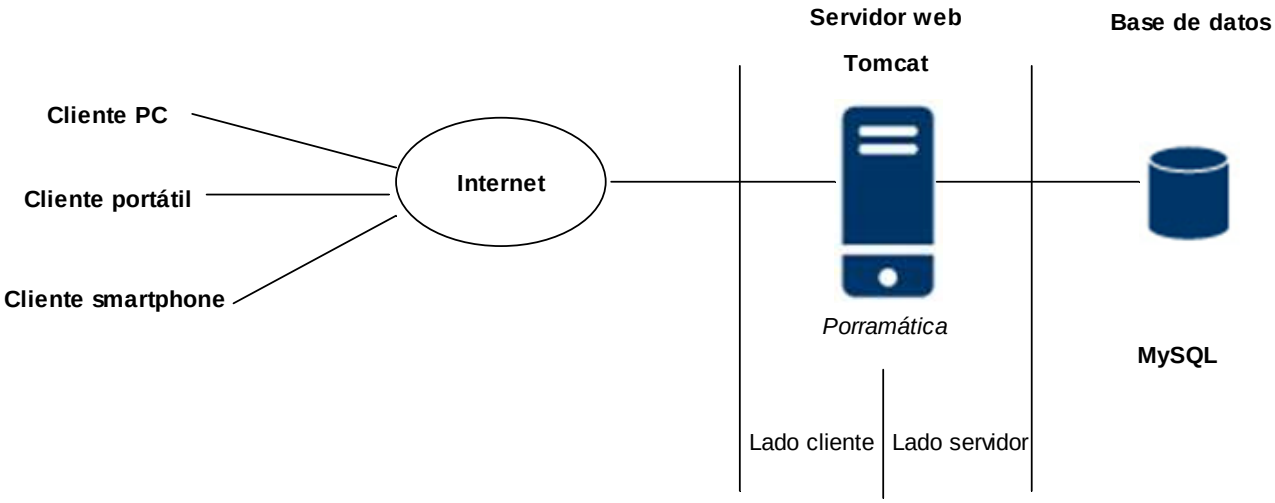

Figura 3.1. Arquitectura Porramática

3.2.2 Arquitectura modular de la aplicación

Debido a que la aplicación ha sido desarrollada en GWT, y el lenguaje de programación es Java, adquirimos una de sus propiedades más interesantes, que es la programación orientada a objetos.

Con ella conseguimos modularizar la aplicación para que el mantenimiento, detección de errores y ampliación sea mucho más rápida que en otro tipo de lenguaje.

Vamos a ver en la siguiente figura cada uno de estos módulos, que permiten la navegación en la aplicación, para posteriormente explicar brevemente en que consiste cada uno (en subsecciones siguientes así como en el manual lo explicaremos con más detalle):

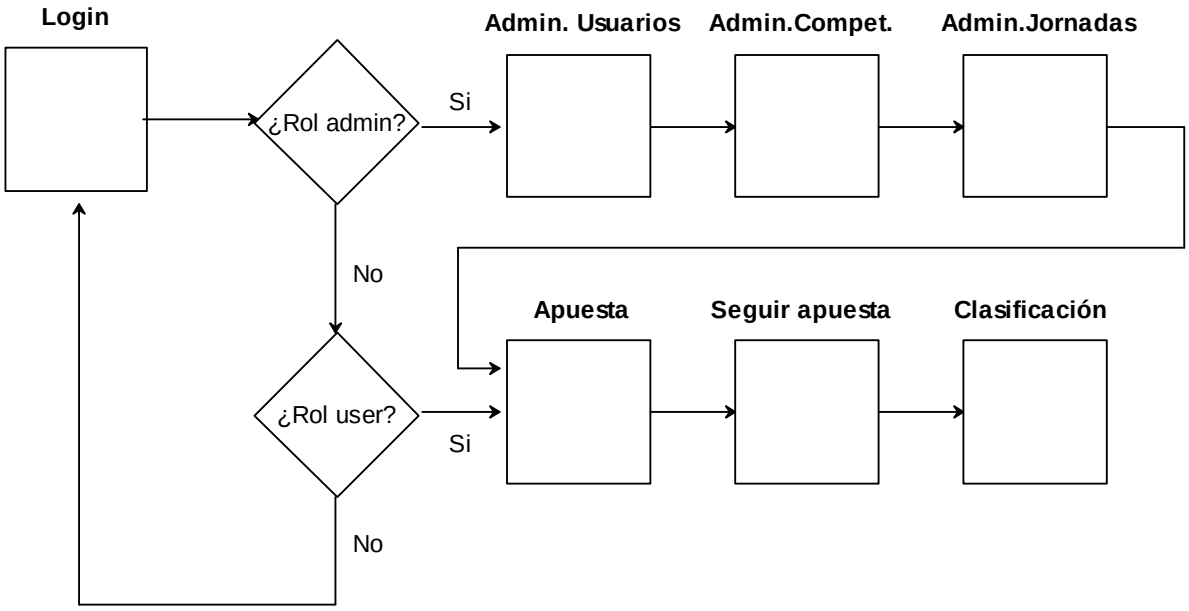

Figura 3.2. Arquitectura modular Porramática

- Ventana de Login. Es la pantalla principal de la aplicación. A través de ella los usuarios entran en la aplicación autenticándose con su usuario y contraseña, ya sean administradores o simplemente usuarios de apuestas. Una vez autenticado el usuario, pasa a la pantalla principal de selección de opción.

Las ventanas que sólo puede ver un usuario administrador son:

- Administración de usuarios. Compuesta de dos pantallas: “Crear usuario” y “Editar usuario”. Este bloque permite al administrador dar de alta nuevos usuarios, permitiéndoles o no la recepción de emails, así como posteriormente la

edición de alguno de los campos de su cuenta o el borrado de dicho usuario de la aplicación.

- Administración de competiciones. Compuesta de las pantallas "Crear competición" y "Editar competición". En estas pantallas, el administrador puede dar de alta una nueva competición para posteriormente añadir jornadas a ella para poder empezar a apostar. También se pueden editar algunos campos de la competición una vez creada, así como borrarla, eliminando de esta manera todas las jornadas adyacentes y las apuestas realizadas sobre las mismas.

- Administración de jornadas. Otras dos pantallas forman este bloque, "Crear jornada" y "Editar jornada". Podemos crear una jornada totalmente configurable en cuanto a puntos a jugar, establecer el tipo de jornada, partidos que se disputarán en ella. En cuanto a la opción de editar, podremos ir modificando los resultados según se vayan sucediendo, o también poder borrar la jornada así como las apuestas que se hayan realizado.

Y ahora se explican las ventanas que pueden visualizar tanto los administradores como los usuarios normales de apuestas:

- Apuesta. Ventana para crear una apuesta. Permite apostar sobre una jornada. Es la funcionalidad final de *Porramática*, permitiendo también la apuesta de un resultado por cada partido jugado.

- Seguir apuesta. Una vez apostado a una jornada, en esta nueva ventana podremos seguir nuestras apuestas y compararlas con lo que está sucediendo en la jornada. Compararemos nuestra apuesta con los resultados en directo, y podremos escrutarlas para esclarecer cuantos aciertos llevamos en ese preciso instante.

- Clasificación. Esta ventana permite al usuario reflejar en que posición está situado respecto a los demás participantes. Se puede ver la clasificación por jornada jugada, o por competición entera, muy útil en el caso por ejemplo de una Eurocopa o un Mundial en el que varios usuarios participan para ver quien consigue acertar más partidos y ser el vencedor del torneo.

Como ya hemos comentado, se explicará con más detalle las configuraciones de cada una de estas pantallas para su total manejo y compresión.

3.2.3 Arquitectura de la base de datos

Viendo todas las opciones que soporta *Porramática* necesitamos una base de datos robusta y fiable que absorba tal cantidad de información que se puede manejar. En la siguiente figura mostramos el diagrama de la base de datos:

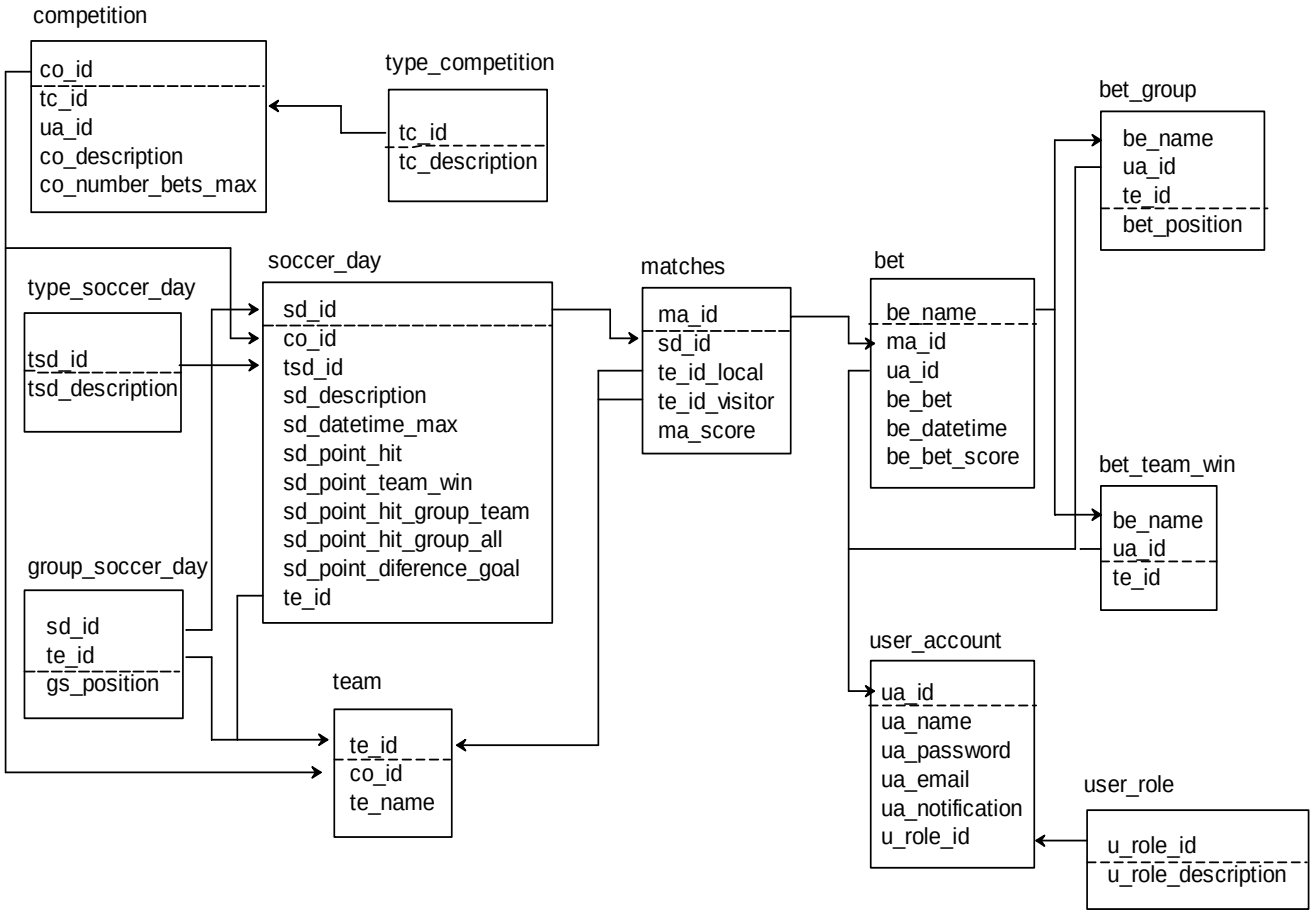

Figura 3.3. Diagrama base de datos

Vamos a explicar en detalle que función sirve cada tabla:

- competition. Tabla que refleja una competición. Posee un tipo de competición, un usuario que ha sido el creador, una descripción y el número máximo de apuestas a realizar sobre dicha competición por jornada.

- type_competition. Tipo de competición. En Porramática puede haber inicialmente cinco tipos de competiciones, pero se puede tanto ampliar como reducir su número modificando la base de datos.

- soccer_day. Refleja una jornada. Esta jornada tiene asignada una competición, un tipo de jornada, una descripción meramente informativa, una fecha máxima de apuesta, que una vez superada dicha apuesta no se permite seguir apostando, una serie de parámetros de puntos autoconfigurables dependiendo de los requisitos del administrador, y el equipo ganador de la jornada si es que lo hay.

- group_soccer_day. Es una tabla que refleja el orden de los equipos en un grupo. Se usa para una jornada que es de tipo copa como por ejemplo una Eurocopa o la Champions League. Esta tabla se modificará cuando el administrador refleje los cambios que se suceden en el grupo en cuestión. El campo “gs_position” es la posición del equipo representado por “te_id”.

- team. Tabla para representar un equipo. Al crear una competición debemos establecer los equipos que participarán en ella. Tiene la clave externa apuntando a la competición, y un campo descriptivo con el nombre del equipo.

- type_soccer_day. Establece los tipos de jornadas disponibles en la aplicación. Se puede crear una jornada de tipo "Partidos" (donde no hay grupos ni equipo ganador), "Grupos" (para rondas clasificatorias donde hay grupos de equipos), y "Grupos y final" (para por ejemplo una competición entera con grupos clasificatorios con un equipo ganador final).

- matches. Esta tabla representa un partido a jugar de una jornada. Tiene como campos la jornada asignada, los equipos que van a jugar clasificados como local y visitante, y por último el marcador, que se irá actualizando a medida que el administrador lo considere oportuno.

- bet. Representa una apuesta de un usuario sobre un partido en concreto. Para ello se hace referencia en los campos al partido en cuestión así como al usuario, los dos con sus claves. Los demás campos son el nombre asignado a la apuesta, el signo quinielístico de la apuesta, el timestamp con la que se realizó el registro de la apuesta, y la apuesta del marcador para el partido.

- bet_group. En este caso es la tabla para la apuesta sobre un grupo. Como hemos mencionado anteriormente, para el caso de ser un tipo de jornada de grupo o grupo y final, podemos apostar sobre el orden de los equipos en el grupo. Esta tabla representa esa apuesta del grupo, por los campos sobre la apuesta, el usuario que realiza la apuesta, y el equipo y posición en que queda dentro del grupo.

- bet_team_win. La otra apuesta que se puede hacer es el equipo ganador. Si la jornada es de tipo "Grupo y final", el usuario debe indicar cuál cree que será el equipo ganador de la competición.

- user_account. Cuando un administrador efectúa el alta de un usuario en la aplicación, queda registrado en esta tabla. Los campos que contiene son su nombre, contraseña de acceso, dirección de email, posibilidad de recepción de alertas a su dirección de correo electrónico y el tipo de usuario que es.

- user_role. Y por último, esta tabla representa el tipo de usuario. Como ya hemos visto a lo largo de esta memoria, la aplicación soporta dos tipos de usuario: administradores, que pueden realizar unas acciones extra con respecto a los usuarios normales, que son el segundo y último tipo de usuario disponible, capaces de efectuar apuestas, seguirlas y ver la clasificación.

3.3 Interfaz de usuario

La interfaz web son los elementos gráficos y estructurales que permiten al usuario acceder a los contenidos, navegar e interactuar. Para lograr que un usuario se sienta

cómodo, permanezca en la página y vuelva más adelante, el diseño de la interfaz es importante.

La facilidad y la comodidad con que los usuarios acceden a los servicios que brinda una web está fundado en dos principios fundamentales para el desarrollo de una interfaz efectiva: la **simplicidad** y la **coherencia**.

- La **simplicidad** con que se desarrolle la interfaz es crucial para determinar que un usuario se sienta satisfecho y desee regresar a un sitio. El hecho de que una persona tenga que realizar una extensa navegación por el sitio para encontrar lo que busca es totalmente contraproducente. Por el contrario, si el sitio dispone de herramientas que permitan acceder rápidamente a aquello que necesita, seguramente volverá. La simplicidad está dada por varios factores a tener en cuenta. El primer concepto importante es que los elementos gráficos o textuales que componen la interfaz deben ser claros y de fácil identificación. Por ejemplo, en *Porramática* se mantiene todo el diseño de la cabecera y pie de página en todas las páginas de la aplicación, como se podrá comprobar en sucesivas capturas de pantalla. De igual manera, el menú lateral para acceder a las distintas secciones siempre permanece visible, por lo que sólo cambia el contenido central para mostrar al usuario la opción seleccionada. O el logout de la aplicación se encuentra situado en la parte superior derecha de la aplicación, a un solo click de distancia independientemente de la sección en que estemos.

- También es importante que se guarde **coherencia** entre los diferentes elementos que componen la aplicación o sitio web, como los títulos, botones, menús emergentes, etc. Todos los elementos que permiten al usuario identificar y navegar deben ser coherentes con el cometido que desempeñan, de forma que la comprensión y búsqueda de los contenidos sean accesibles por el usuario sin que tenga la necesidad de realizar complejos razonamientos. En *Porramática* hemos organizado las secciones mediante un menú lateral, para que el usuario sepa en todo momento en que lugar se encuentra, y a su vez si necesita volver a otra opción lo tenga siempre accesible y en a un solo click de distancia. De igual forma se guarda el mismo estilo de los elementos en todas las secciones que dispone la aplicación para facilitar su reconocimiento.

A la hora de abordar el diseño de la interfaz de la aplicación se ha intentado que fuera una aplicación sencilla de fácil uso tanto para los usuarios como para los administradores, permitiendo que sin necesidad de tener conocimientos técnicos, su manejo y uso sean intuitivos y accesibles.

3.3.1 Diseño orientado al usuario

Como hemos comentado anteriormente, un punto importante a la hora de diseñar una aplicación web es la simplicidad. De esta manera al usuario le agrada acceder a la aplicación y le invita a hacerlo a menudo porque consigue lo que quiere hacer en poco tiempo. Esto lo conseguimos con el menú lateral que posee la aplicación. El usuario tiene a un solo click de distancia las tres opciones que puede realizar en la aplicación.

En la siguiente figura podemos ver el menú del usuario:

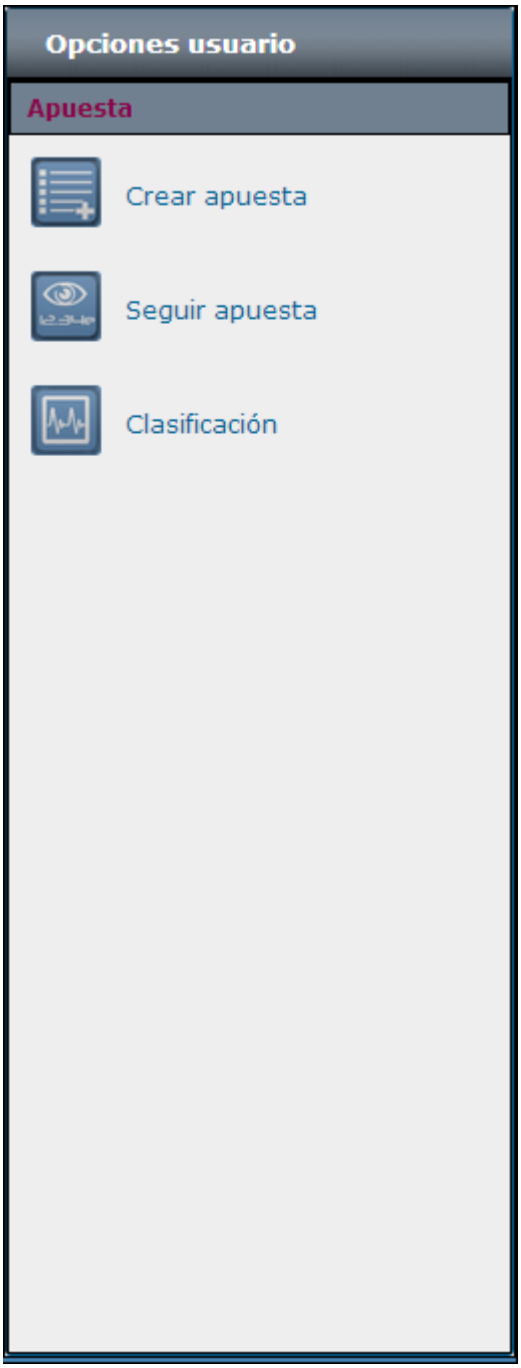

Figura 3.4. Menú del usuario

3.3.2 Diseño orientado al administrador

De igual forma, el administrador posee un menú muy parecido al del usuario, pero con un número mayor de opciones a manejar. Podemos ver en la siguiente figura el menú del administrador en sus dos versiones desplegadas:

Figura 3.5. Menú del administrador

Se observa como el menú se contrae dependiendo de las opciones que queramos visualizar, agrupándolas en “Usuarios” y “Apuesta”. De esta forma se puede navegar por la aplicación muy rápidamente a un solo click de distancia entre secciones.

Las opciones del administrador son más variadas que las opciones que tiene un usuario. Esto lo vimos en la arquitectura modular de la aplicación, y lo volvemos a confirmar aquí viendo los menús disponibles para cada tipo de usuario.

3.4 Desarrollo de la aplicación

3.4.1 Registro de cuenta de correo

Para que la aplicación pueda enviar correos electrónicos, debemos poseer una cuenta de correo. Nos decantamos por GMail como ya comentamos en la subsección de JavaMail. Para ello debemos crear una cuenta para usar el servicio de Google y seguir los pasos que nos describen. [19]

En nuestro caso la cuenta de correo es porramatica@gmail.com. Esta es la dirección remitente que verán los usuarios cuando reciban los correos electrónicos de la aplicación.

Con esto ya tenemos creada nuestra cuenta de GMail.

3.4.2 Registro de cuenta de Twitter

La aplicación es capaz de tuitear la información relacionada con las jornadas. Pero para ello necesitamos una cuenta en Twitter. En el siguiente enlace podremos crear una cuenta en dicha red social. [20]

Para nuestra aplicación, usamos la dirección de correo de GMail creada anteriormente, y con ello conseguimos registrarnos en Twitter con el nombre de usuario @porramatica.

El perfil que usa nuestra aplicación es @porramatica. Todos los tuits que publique la aplicación van a ser públicos por lo que cualquier usuario puede verlos.

Un usuario de *Porramática* que quiera usar todo el potencial de la aplicación para seguir los resultados en directo y ver cuando puede realizar su apuesta, al crear su usuario marcará su casilla de habilitación de envío de emails, y a su vez y si tiene perfil en Twitter, puede seguir a *Porramática* en @porramatica para tener un control total sobre esta.

3.4.3 Registro de aplicación twittermail

Pero, ¿cómo consigue Porramática tuitear información en su perfil? Utilizando el módulo de la aplicación que envía emails (visto en el API de JavaMail en la subsección de Java) investigamos esta aplicación: **twittermail** [18]

Twittermail nos permite tuitear en nuestro perfil de Twitter al enviar un email desde la dirección con la cual nos registramos en Twitter (porramatica@gmail.com) a la dirección que nos otorga la aplicación, del estilo usuario_twitter@twittermail.com, en nuestro caso porramática@twittermail.com.

Por lo tanto, cada vez que Porramática envía un email, este automáticamente es tuiteado en el perfil de Twitter.

Para poder usar esta aplicación debemos dar permisos a dicha aplicación a nuestra cuenta de Twitter. Para ello debemos dirigirnos a la página de twittermail e introducir nuestro usuario de Twitter, con lo que nos aparecerá la siguiente ventana:

Figura 3.6. Registro de twittermail

Como vemos en la imagen, nos pide que autoricemos la aplicación en nuestra cuenta de Twitter. Podemos observar como arriba a la derecha aparece nuestra cuenta ya que estamos logueados en ese instante, sino nos aparecerán los campos para introducir nuestro usuario y contraseña de Twitter.

Una vez dado permiso a la aplicación, ya podremos tuitear información en nuestro perfil de Twitter mandando un email desde la dirección de correo de la aplicación.

3.4.4 Diseño de la interfaz de Porramática

En la siguiente figura podemos ver la interfaz de la aplicación cuando accedemos a ella. Consta de una cabecera con el nombre de la aplicación, un pie de página

manteniendo el estilo, y el cuerpo formado por un formulario para insertar el usuario y contraseña de acceso a la aplicación.

Figura 3.7. Ventana loguin Porramática

Ahora que hemos visto la ventana de loguin, ¿cómo consigue la aplicación el logueo de un usuario? Vamos a explicar en detalle el RPC de GWT en este apartado, por lo que esta llamada remota es extrapolable a cualquier otra llamada que se realice en otra ventana de la aplicación. Pero antes de explicar en detalle la llamada remota de loguin vamos a ver las clases que maneja la aplicación para el manejo de las llamadas remotas.

En nuestro caso disponemos de un servicio llamado PorraWebRemoteService. Esta interfaz es la que define todos los métodos remotos que podemos usar en la aplicación. Es lo que en la figura 2.6 vemos como “YourService (interface)”. Como vemos, extiende de RemoteService.

Por otro lado tenemos PorraWebRemoteServiceAsync, que es otra interfaz exactamente igual que la anterior pero que los métodos descritos en ella reciben al final un parámetro adicional que es el retorno del método a implementar.

Por último tenemos la clase PorraWebRemoteServiceImpl que como vemos en la figura extiende de RemoteServiceServlet e implementa obviamente nuestra interfaz primera para que aquí si desarrollemos los métodos en cuestión.

Y a continuación explicamos como se realiza la llamada remota para que un usuario pueda loguearse con la aplicación:

- El usuario rellena los campos “Usuario” y “Contraseña” y pulsa sobre el botón “Iniciar sesión”.

- La clase PorraWebAccessController recibe la llamada del **lado cliente** y la redirige al servidor, por lo que implementa un success por si la ejecución de la llamada es correcta o un failure por si hay problemas de conexión. La llamada se redirige al servidor por medio de la interfaz remota.

- Llega la llamada al **lado servidor**. El método en cuestión accede a la base de datos y comprueba si el usuario está en ella con esa contraseña. Hay que notar que el servicio de base de datos devuelve un objeto usuario del lado del servidor, por lo que hay que serializarlo al lado del cliente.

- La llamada es devuelta a la clase PorraWebAccessController con la respuesta del servidor. Si la llamada ha sido correcta se procede tanto si el usuario está logueado mostrándole la pantalla de inicio, como si no lo está mostrándole un mensaje de error.

Una vez logueados accedemos a la página principal de toda la aplicación. Como se puede ver en la imagen de la figura 3.8, en la parte superior derecha se muestra el nombre del usuario junto con la opción de “Salir” de la aplicación para volver a la ventana de loguin.

Desde la aplicación se muestra ocupando la parte izquierda de la pantalla el menú desde el cual se accede a cualquier sección de *Porramática*. Dependiendo del tipo de usuario que se haya logueado, podemos ver que el menú es encabezado por “Opciones administrador” u “Opciones usuario”. Ambas figuras las reflejamos en 3.4 y 3.5.

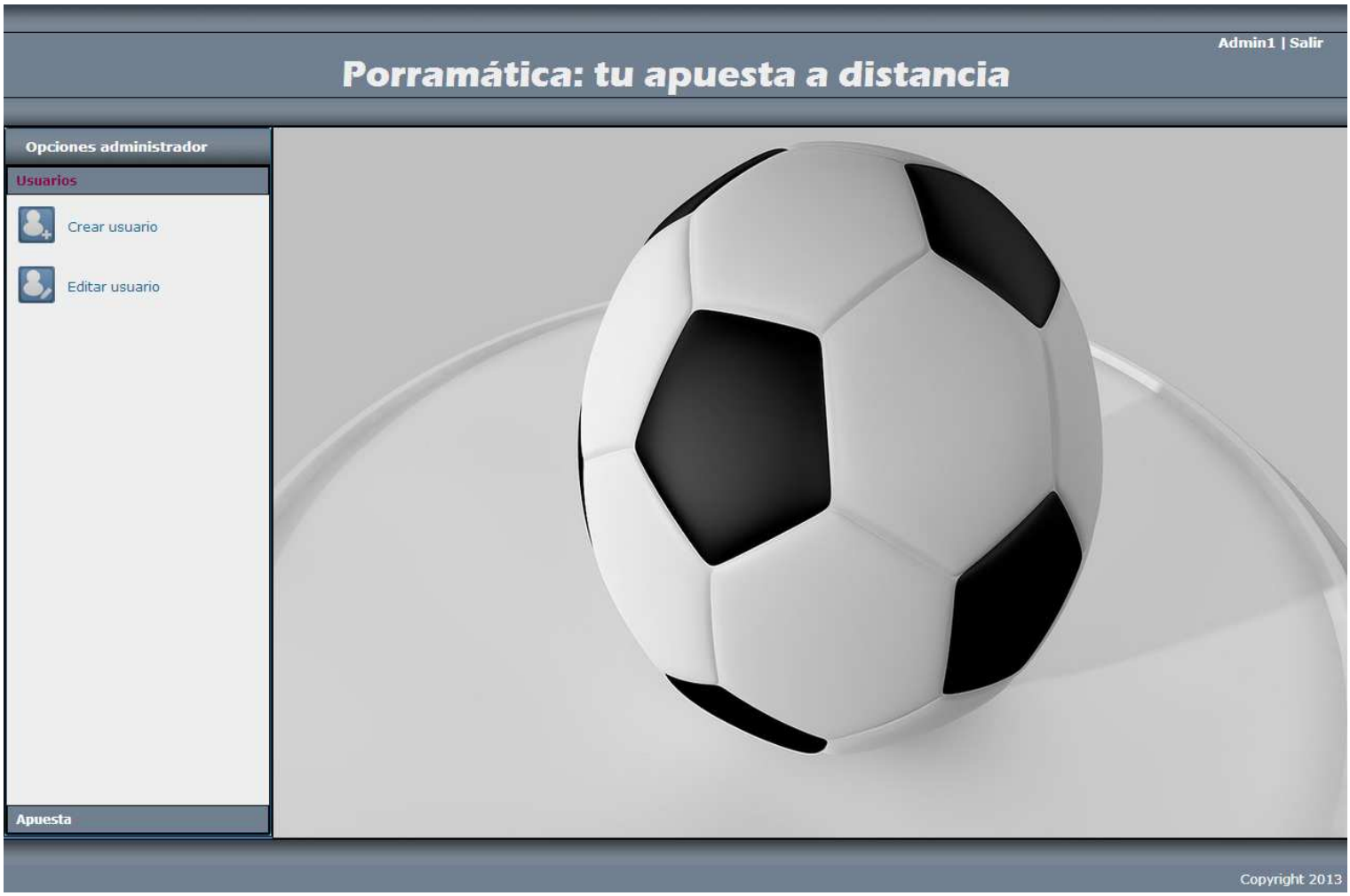

Figura 3.8. Ventana principal Porramática

3.4.5 Desarrollo de los módulos funcionales

Como ya hemos comentado en subsecciones anteriores, tenemos dos formas de autenticarnos en la aplicación: siendo administradores o usuarios normales. Dependiendo de que rol de usuario estemos logueados, aparecerán unas funciones u otras en *Porramática*. Es lo que denominamos como módulo funcional.

A continuación explicaremos cada módulo de la aplicación en detalle para que el lector sepa que función realiza cada módulo, y sirva a modo de manual de la aplicación.

3.4.6 Administración de usuarios

Este módulo es el encargado de dar de alta usuarios nuevos en la aplicación; así como poder editarlos o eliminarlos de ella.

Para ello nos apoyamos en dos menús que son “Crear usuario” y “Editar usuario”.

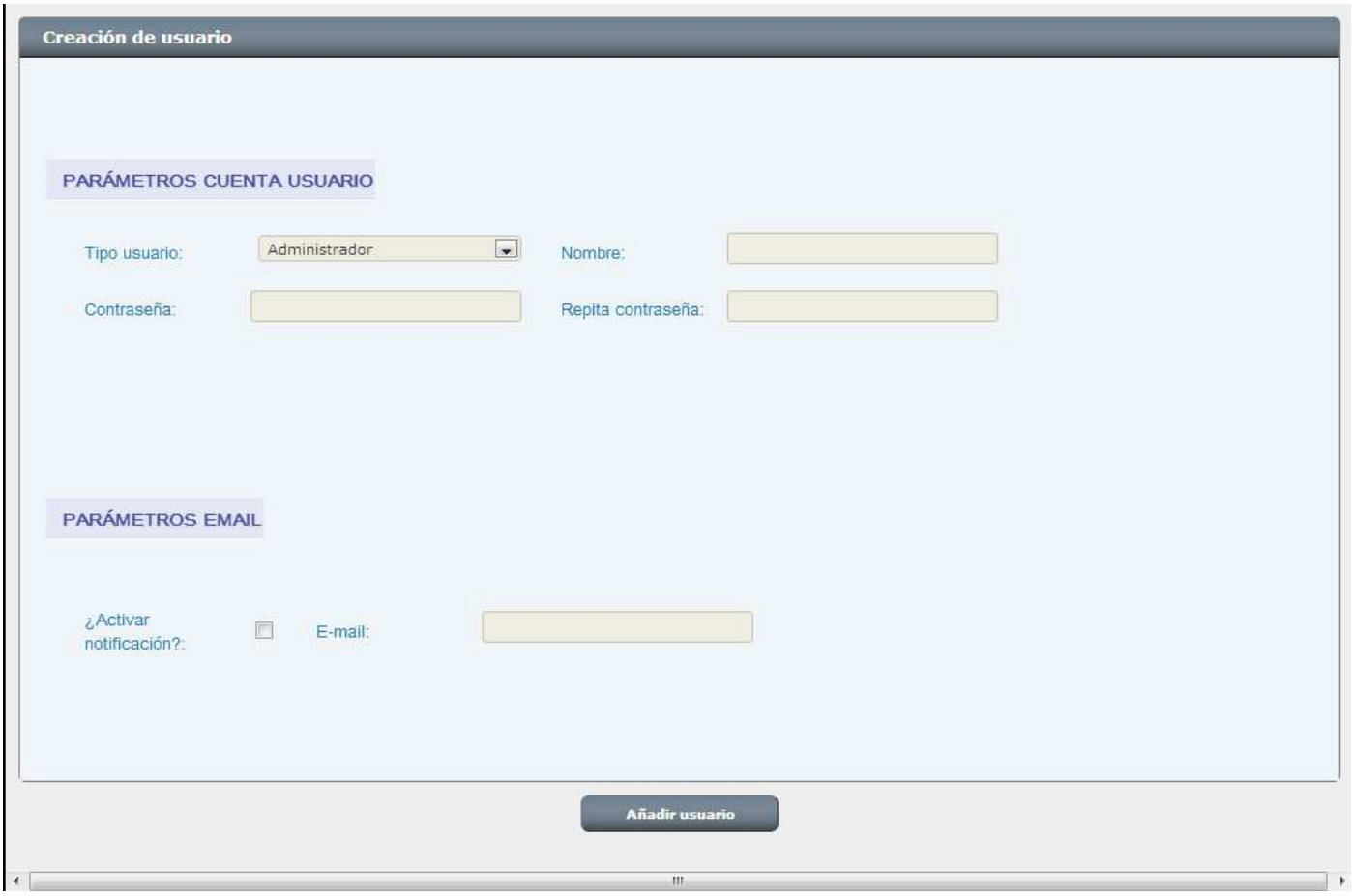

Figura 3.9. Creación de usuario

Como podemos ver es un formulario simple, donde nos pide que seleccionemos el tipo de usuario a crear, su nombre, contraseña, habilitar la recepción de correos electrónicos y por último su propia dirección de correo donde recibirá las alertas. Todos los formularios de la aplicación tienen comprobación de errores como no dejar ningún elemento en blanco o comprobación de formato de los campos.

Cuando hayamos rellenado los datos, sólo tenemos que pulsar “Añadir usuario” para darle de alta en el sistema.

En cuanto a la edición, podemos ver en la figura 3.10 que varía un poco con respecto al formulario de alta de usuario. Este posee una lista de elementos en los cuales se agrupan los usuarios. Si seleccionamos un elemento, es decir, un usuario, automáticamente se rellenan los campos de dicho usuario. De esta forma podremos editar uno o varios campos a la vez.

También tenemos la opción de borrar el usuario seleccionado del sistema. En este caso se nos pedirá, en un popup de confirmación, si realmente deseamos borrarlo. En caso afirmativo, el usuario quedará borrado del sistema.

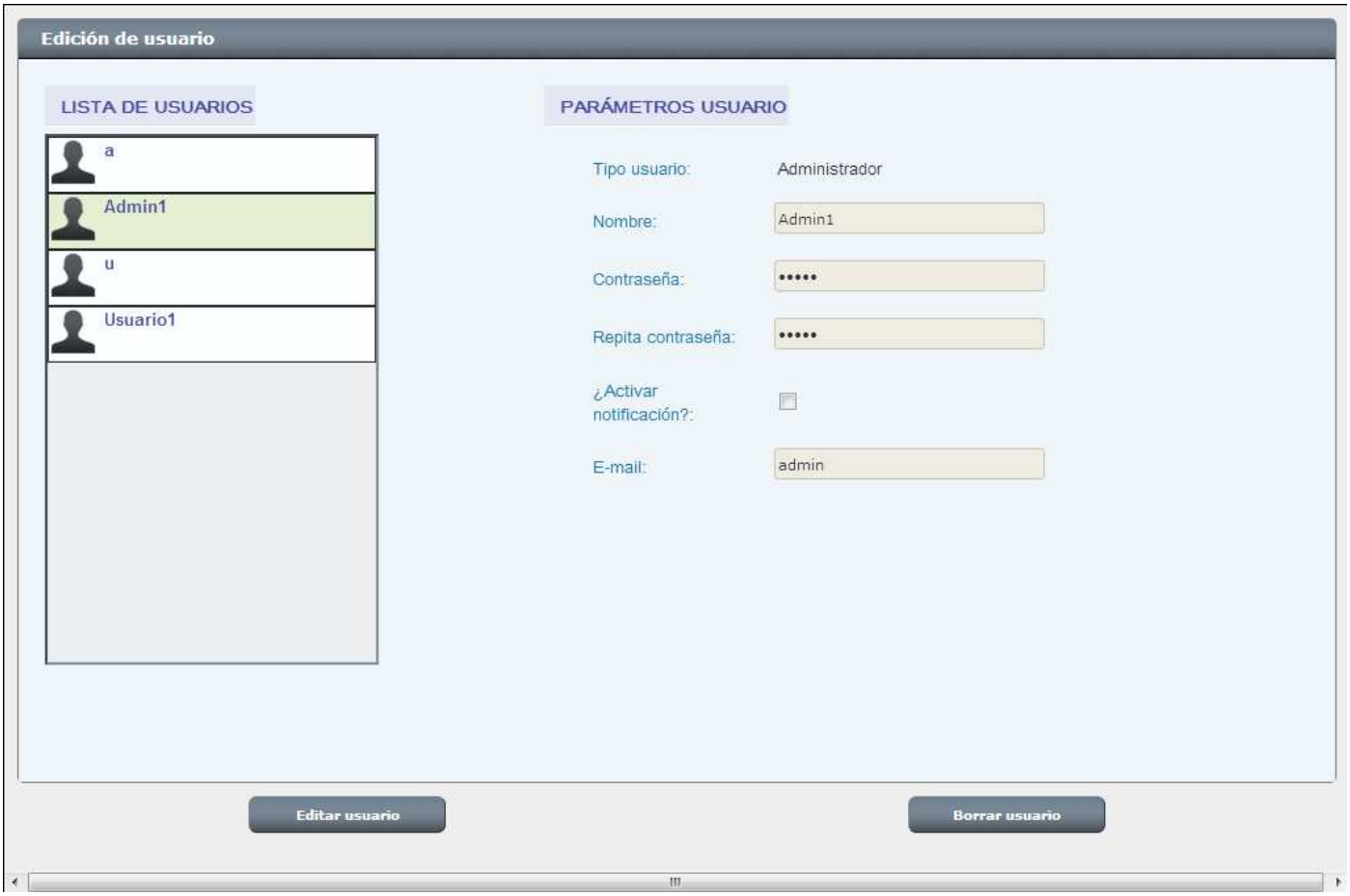

Figura 3.10. Edición de usuario

3.4.7 Administración de competiciones

Para poder crear una jornada de juego, primero hay que crear una competición donde residirán las jornadas asociadas. Para ello disponemos del módulo de "Crear competición". En la figura 3.11 podemos ver el diseño de esté módulo.

Podemos diferenciar dos partes. La primera consta de los parámetros de la competición tales como el tipo de competición a jugar, su descripción, el usuario que va a crearla y por último el número máximo de apuestas que acepta una jornada para esta competición.

La segunda parte es donde insertaremos los equipos participantes de la competición. Como podemos ver hay dos botones señalados por los iconos más y menos. De esta manera podemos añadir o eliminar un equipo de la competición.

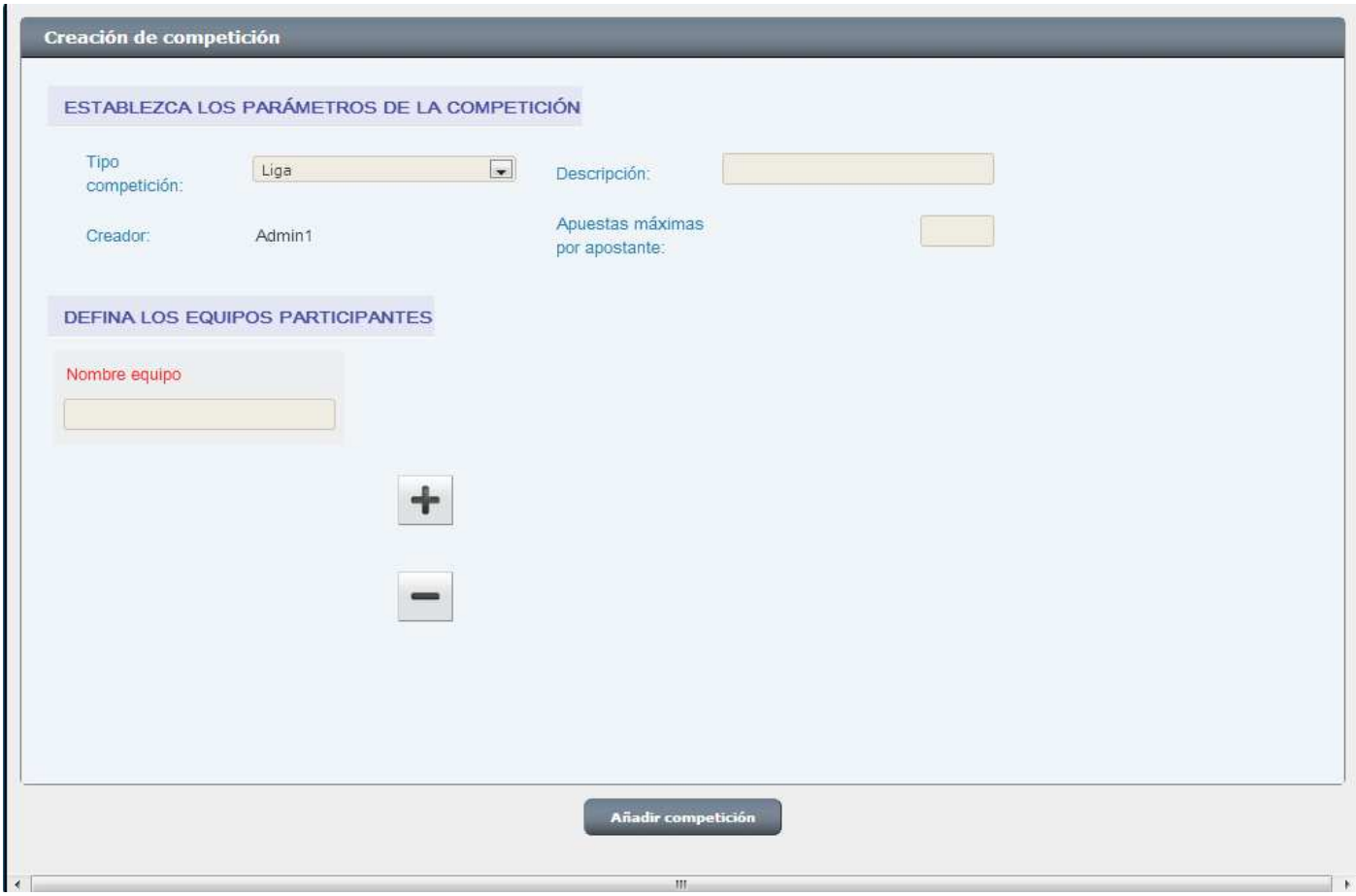

Figura 3.11. Creación de competición

Este módulo también está compuesto por la edición de una competición. Para ello, en la ventana de "Editar competición" disponemos de un elemento de tipo listbox en el cual al pinchar en el nos desplegará una lista de competiciones disponibles.

Podemos editar la descripción de una competición o el número de apuestas máximas a jugar por jornada. Y para terminar también podemos eliminar la competición seleccionada pulsando el botón de "Borrar".

3.4.8 Administración de jornadas

Vamos a explicar a continuación la última opción exclusiva que tiene un administrador. Como en el anterior módulo tenemos dos opciones: "Crear jornada" y "Editar jornada".

La creación de jornada es la que mostramos en la siguiente figura:

Figura 3.12. Creación de jornada

Para crear una jornada debemos seleccionar la competición en la cual irá enmarcada. A continuación debemos seleccionar el tipo de jornada. Como comentamos en secciones anteriores puede ser de tipo “Partidos”, donde sólo habrá puntos por acertar signo quinielístico y por acertar el resultado (la puntuación por acertar el resultado sólo se suma en caso de acertar el resultado completo; dicha puntuación se aplica como la suma de los goles del encuentro por la puntuación de este campo); “Grupo” donde se deberá elegir el número y equipos que forman el grupo y se suman las puntuaciones por acertar la posición del equipo en el grupo y la puntuación por acertar todas las posiciones de los equipos en el grupo; y “Grupo y final” con mismas características que el “Grupo” pero con puntos por acertar el equipo ganador de la competición.

Estableceremos una descripción para la competición y una fecha máxima de apuesta. Esta fecha máxima es para controlar que ningún usuario puede apostar después de las 23:59:59 horas del día que marcamos en esa casilla.

Sólo nos queda por establecer los equipos que forman la jornada y los puntos que asignamos por cada característica de la jornada.

En cuanto a la edición de jornada, en esta ventana el administrador podrá editar los resultados de los partidos a medida que se vayan sucediendo, el orden del grupo si es que la jornada es de tipo “Grupo” o “Grupo y final”, ó el equipo ganador de la competición en caso de saber ya el equipo ganador de la competición. En la figura 3.13 se puede ver la

ventana de edición de jornada para ver la disposición de los elementos, en este caso para una jornada de liga, es decir, de “Partidos”.

Figura 3.13. Edición de jornada

3.4.9 Apuesta

Este módulo es común tanto para los administradores como para los usuarios, y al fin al cabo es para lo que está diseñada esta aplicación. Desde aquí podremos apostar por las diferentes jornadas que estén dadas de altas por el administrador, o por aquellas que no hayan superado aún la fecha máxima de apuesta.

Como podemos ver en la figura 3.14 el usuario dispone de un campo para establecer un nombre a su apuesta. También se le informará del número de apuestas que aún le quedan por hacer para la jornada que está seleccionando.

Se le pedirá que introduzca el signo quinielístico y los resultados para los partidos de las jornadas. También se le puede pedir que establezca su orden de grupo para la jornada en caso de ser una jornada de tipo “Grupo” o “Grupo y final”, y por supuesto el equipo ganador de la competición en caso de ser de este último tipo.

Para crear la apuesta debemos pulsar en el botón “Añadir apuesta”. En caso de que haya incorrección de datos o algún formato equívoco de algún campo la aplicación nos lo notificará mediante un mensaje de texto.

Figura 3.14. Crear apuesta

3.4.10 Seguir apuesta

Explicamos en esta subsección como el usuario puede seguir sus apuestas. Para ello en la ventana de “Seguir apuesta” podemos seleccionar en que jornada están las apuestas que queremos seguir. Una vez seleccionado nos aparecerán los partidos que se están jugando en dicha jornada, con los últimos resultados actualizados por el administrador.

A continuación si el usuario selecciona una de sus apuestas para la jornada seleccionada, se rellenarán sus resultados y sus signos quinielísticos.

Para facilitar una mejor lectura de los aciertos del usuario, se ha introducido un botón inferior con el título “Escrutar” en el cual se procesa la información para hacer el escrutinio de la jornada con la apuesta del usuario. Se abrirá una ventana emergente donde se podrá leer los aciertos del usuario, los puntos que lleva acumulados, si hay partidos de grupos cuantas posiciones lleva acertadas, etc.

Mostramos en las siguientes figuras como es el módulo de seguir apuesta y una ventana de escrutinio para la apuesta seleccionada.

Figura 3.15. Seguir apuesta

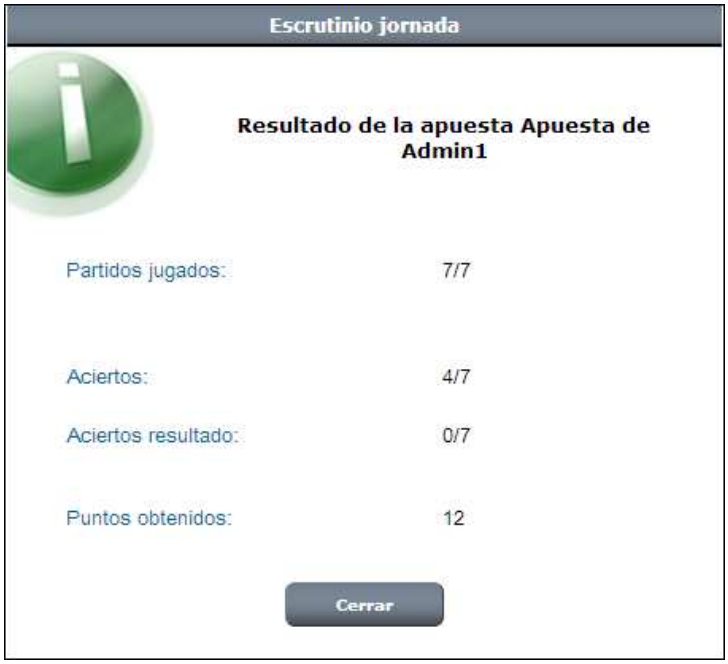

Figura 3.16. Escrutinio apuesta

3.4.11 Clasificación

Y por último explicamos el módulo funcional de la clasificación. Podemos clasificar las puntuaciones de los usuarios por dos métodos: por jornada o por competición. Si seleccionamos la opción de jornada, nos aparecerá la clasificación con las apuestas realizadas sobre esa jornada y la puntuación de cada usuario con la descripción de la apuesta en sí.

En cambio si seleccionamos la clasificación por competición, se realizará una clasificación donde se ordenará de mayor a menor los usuarios según los puntos obtenidos en cada una de las jornadas disputadas, haciendo un cómputo de todas ellas.

Hemos incluido un botón de "Refrescar" para que el usuario no tenga que estar seleccionando de nuevo la opción que quería visualizar, y de esta manera es mucho más cómodo el refresco de la página.

Figura 3.17. Clasificación

3.5 Manual de usuario

Este apartado sirve como manual de usuario para el uso de la aplicación *Porramática*. No se requieren conocimientos previos de programación para su utilización, tan sólo un uso familiarizado con la web para reconocer formularios típicos de cualquier página.

En los subapartados siguientes, explicaremos paso a paso como utilizar la plataforma tanto para los usuarios administradores como para los usuarios que únicamente pueden jugar sus apuestas.

3.5.1 Requisitos

Es necesario disponer de una conexión a Internet para poder acceder a *Porramática*. También es necesario tener una cuenta en la aplicación. Si no la tiene debe ponerse en contacto con un administrador y solicitarle acceso a la misma.

Una vez disponga de un usuario y contraseña, en la ventana de loguin introduzca sus credenciales y entrará en la aplicación.

3.5.2 Manejo por administradores

Una vez logueado como administrador le aparecerá la ventana principal. En ella puede elegir cualquier opción que dispone en la parte central izquierda, así como salir de la aplicación mediante en enlace situado en la parte superior derecha de la aplicación. Puede elegir cualquier función de entre las que posee la aplicación, que se pueden visualizar en la figura 3.2.

Para profundizar en cualquier opción de la aplicación le aconsejamos que visite la sección 3.4 de esta memoria.

3.5.3 Manejo por usuarios

El resto de usuarios, al igual que los administradores, al entrar en la aplicación les aparecerán las opciones de que dispone en la aplicación. Pulse cualquier opción del menú lateral izquierdo para abrir dicha función.

Para profundizar en cualquier opción de la aplicación le aconsejamos que visite la sección 3.4 de esta memoria.

Si desea salir de la aplicación sólo debe pulsar la opción “Salir” situada en la parte superior derecha de la pantalla.

Capítulo 4

Pruebas

Durante el desarrollo de este proyecto se han realizado diferentes pruebas sobre la aplicación. En las siguientes secciones se describe en qué consiste cada escenario de prueba y los resultados obtenidos. También se especificará una sección donde demostraremos el funcionamiento de la aplicación en un caso real de competición.

4.1 Escenarios de pruebas

Se han empleado dos escenarios de pruebas para la aplicación. A continuación describimos cada uno de ellos.

4.1.1 Pruebas en local

Las pruebas en local son aquellas que realizamos en el propio equipo de desarrollo, para agilizar el mismo de tal manera que no perdamos tiempo en probar la versión en su entorno real por el consiguiente gasto de tiempo que conlleva el producir una versión para desplegar en el servidor web.

Como comentamos en la sección de GWT, este posee un modo de ejecución llamado “Hosted Mode” en el cual podemos ejecutar nuestra aplicación web sin tener que pasar por el paso previo de compilación y traducción de código Java a JavaScript. Tan solo

necesitamos tener los plugins necesarios en nuestro entorno de desarrollo (en nuestro caso Eclipse) y podremos ejecutar este modo.

Algunos de los problemas encontrados en este tipo de pruebas fueron las siguientes, siguiendo un orden cronológico de desarrollo de menor a mayor:

- No soportaba los estilos del archivo PORRAWEB.css. Cuando lanzábamos la aplicación comprobábamos que no aplicaba los estilos que detallábamos en el archivo. Después de consultar la página oficial de GWT corroboramos que al estar utilizando la última versión del framework, la sentencia donde enlazamos el archivo .css debe ir en el archivo PORRAWEB.gwt.xml situado en la parte servidor del directorio del proyecto mediante la sentencia

  <stylesheet src='/PORRAWEB.css'/>

- Seguía sin aparecer la página de estilo habiendo insertado la anterior sentencia. Tuvimos que volver a leer la guía de despliegue de GWT y comprobar que también debe ir la siguiente sentencia en la página HTML de la aplicación:

  <link type="text/css" rel="stylesheet" href="PORRAWEB.css" />

- Instalación de plugin para navegador Firefox. Parece contradictorio, pero parece que Google se ha olvidado de su navegador Chrome para el desarrollo de su propio framework. Este último no permite la instalación del plugin necesario para poder depurar en modo "Hosted" la aplicación. Tuvimos que instalar el plugin y probar toda ella en Firefox. Esto no tiene nada que ver a cuando desplegamos y compilamos, ya que el compilador traduce el código a JavaScript para la mayoría de navegadores web, y luego esta corre como si de una página web se tratase.

- Clase Calendar no soportada en GWT. Una de las cosas que comentamos acerca de GWT es que este no soporta en el lado cliente algunas de las librerías más comunes de Java. Y una de ellas es Calendar. Aunque hay métodos deprecados de la clase Date, tuvimos que utilizarla para suplir la no compatibilidad con Calendar. Para ello usamos un SimpleDateFormat en el lado de servidor que transforma la hora que llega del lado cliente para poder insertarla en la base de datos.

- No se envían emails desde la aplicación y por lo tanto no hay tuiteo en el perfil de Twitter. Es un error por el equipo de desarrollo, ya que tenemos un antivirus que contiene un módulo para controlar y analizar todos los correos electrónicos entrantes y salientes, y dicho antivirus creyó que lo mejor era no recibir ni enviar emails por parte de la aplicación. Se corrigió pausando este módulo para las pruebas.

4.1.2 Pruebas en servidor

Las pruebas en el servidor son las pruebas finales porque vamos a poder comprobar como se comporta la aplicación en el entorno real de ejecución, es decir, en producción,

Ahora sí que tenemos que realizar todos los pasos que proporciona GWT para compilar la aplicación, traducir todo el código a JavaScript, y desplegarlo en el Tomcat.

Asumimos que la base de datos MySQL ya está instalada tanto para las pruebas locales como para estas pruebas, porque sino no habría conexión. Por lo tanto debemos tener correctamente instalado el servidor web Tomcat y realizar correctamente la creación del archivo .war con la aplicación compilada.

Vamos a ver algunos de los problemas que nos encontramos:

- El propio plugin de GWT para Eclipse no compilaba la aplicación. Esto significaba que no podíamos desplegar la versión en Tomcat. Leyendo en los muchos foros que posee Google para GWT, encontramos una solución a nuestros problemas. Un nuevo plugin que a la vez que compila la aplicación, la empaqueta en un archivo .war, haciendo dos pasos en uno. [21]

- No hay conexión con la aplicación. Una vez desplegada la aplicación, probamos a abrirla en un navegador y no podemos visualizarla. Esto es debido a que el PC de producción está en la universidad, y por seguridad debemos conectarnos mediante una VPN. Lógicamente una aplicación de este tipo debe estar abierta al público, pero al estar usando un recurso de la universidad hay que estar protegido frente a males externos y la única solución es probarlo a través de la VPN.

- No hay conexión con la parte servidor. Una vez conseguido desplegar la versión en Tomcat, veíamos que no podíamos loguearnos, a lo cual la aplicación responde con un label indicando el problema. No es problema de loguin, sino de cualquier llamada realizada al servidor. Al ser una aplicación en desarrollo no podemos ver la traza del error como pasa en modo depuración en “Hosted Mode” por lo que tuvimos que recurrir a los logs que genera Tomcat en su directorio. Vimos entonces que la llamada remota no era capaz de encontrar las clases de la parte del servidor. Esto nos hizo pensar que debíamos también exportar los .class de toda la parte servidor al proyecto web. Esta tarea se puede hacer con Eclipse exportando todo el proyecto servidor a un .jar, y una vez hecho esto copiarlo al directorio lib del proyecto web.
  El siguiente paso es volver a compilar y empaquetar la aplicación apoyándonos en el plugin mencionado anteriormente, y esta vez sí la aplicación está correctamente desplegada.

4.1.3 Tablas de pruebas

Vamos a recoger en unas tablas las pruebas más relevantes hechas para cada tipo de escenario de pruebas.

Hemos considerado las pruebas más relevantes para probar la aplicación en su totalidad, así como pruebas que surgen en producción que difieren de las realizadas en modo desarrollo.

En la tabla 4.1 se pueden ver las pruebas en escenario local, y en la 4.2 las pruebas en el entorno real de ejecución.

Prueba realizada	Resultado	Observaciones
Acceso a la aplicación	OK	Accedemos a la aplicación depurando en modo “Hosted”.
Acceso a la base de datos	OK	Al acceder a la aplicación, nos podemos loguear con cualquier usuario, por lo que hay acceso a la base de datos.
Acceso como administrador	OK	El volcado de la base de datos se realiza con un usuario de administrador para poder entrar a la aplicación.
Acceso como usuario	OK	El administrador realiza el alta de un usuario y este puede acceder correctamente a la aplicación.
Administración de competiciones	OK	Todo el módulo funciona a la perfección tanto al crear, editar o borrar.
Administración de jornadas	OK	Al igual que en competiciones, se permiten todas las acciones para las jornadas.
Apuesta	OK	Se realiza una apuesta para usuarios administradores y usuarios normales correctamente.
Seguir apuesta	OK	Podemos ver las apuestas realizadas por un usuario, ver los resultados de los partidos en juego, y hacer un escrutinio de las apuestas seleccionadas.
Clasificación	OK	La aplicación elabora correctamente la clasificación por jornada y por competición.

Tabla 4.1. Pruebas en escenario local

A continuación mostramos una tabla para el escenario del servidor, pero viendo los problemas relevantes y asumiendo que los correctos ya están descritos en la tabla anterior.

Prueba realizada	Resultado	Observaciones
Acceso a la aplicación	ERROR	No visualizamos la aplicación, debido a que el equipo universitario no permite accesos externos. Solución: VPN.
Acceso a la base de datos	ERROR	No hay conexión con la parte servidor. Encontramos el error en que no encuentra las clases del servidor. Arreglado.
Acceso como administrador	ERROR	No hay conexión con la parte servidor. Encontramos el error en que no encuentra las clases del servidor. Arreglado.
Acceso como usuario	ERROR	No hay conexión con la parte servidor. Encontramos el error en que no encuentra las clases del servidor. Arreglado.
Administración de competiciones	OK	Todo el módulo funciona a la perfección tanto al crear, editar o borrar.
Administración de jornadas	OK	Al igual que en competiciones, se permiten todas las acciones para las jornadas.
Apuesta	OK	Se realiza una apuesta para usuarios administradores y usuarios normales correctamente.
Seguir apuesta	OK	Podemos ver las apuestas realizadas por un usuario, ver los resultados de los partidos en juego, y hacer un escrutinio de las apuestas seleccionadas.
Clasificación	OK	La aplicación elabora correctamente la clasificación por jornada y por competición.

Envío de emails	ERROR	Problema por el antivirus instalado que no permitía el envío de emails. Arreglado
Tuiteo en Twitter	ERROR	Problema subsanado al poder enviar emails por parte de la aplicación.

Tabla 4.2. Pruebas en servidor

4.2 Demostración competición real

En este apartado vamos a ilustrar al lector con la configuración de una competición real, en este caso la Eurocopa de fútbol de 2012. Veremos como el administrador realiza la configuración de la misma, y posteriormente realizará el alta de cada una de las jornadas que la componen.

Posteriormente los usuarios podrán empezar a apostar sobre las jornadas dadas de alta, podrán seguir sus apuestas individualmente y escrutarlas, o también consultar la clasificación general.

4.2.1 Escenario de juego

La competición que vamos a configurar y a mostrar es la Eurocopa de fútbol de 2012. La competición se estructura en una fase de previa en la que hay cuatro grupos (del A al D) cada uno de ellos con 4 equipos. En cada grupo se enfrentará cada equipo con los 3 restantes, por lo que en cada grupo se disputan 6 partidos.

Los dos primeros clasificados de cada grupo pasan a la fase de cuartos de final, por lo que nos encontraremos con 8 equipos finales. Se disputan 4 partidos de cuartos para dar paso a los 4 mejores equipos que lucharán en semifinales en otros 2 partidos. De dichas semifinales saldrán los dos equipos finales que disputarán la final y conoceremos al ganador del campeonato.

Vamos a contar con la presencia de 3 participantes para esta demostración. Un administrador (Admin1) que se encargará de dar de alta la competición y las jornadas pertinentes. Y otros dos usuarios (User1 y User2) que podrán apostar sobre las jornadas a disputar.

Mostraremos las dos primeras jornadas a jugar correspondientes al grupo A y B, para luego pasar a cuartos de final, semifinal y final, para de esta manera mostrar al lector cada una de las posibilidades que tiene *Porramática*.

4.2.2 Configuración competición

En primer lugar el administrador es el encargado de dar de alta la competición. Podemos ver en la siguiente figura como se procede con este paso:

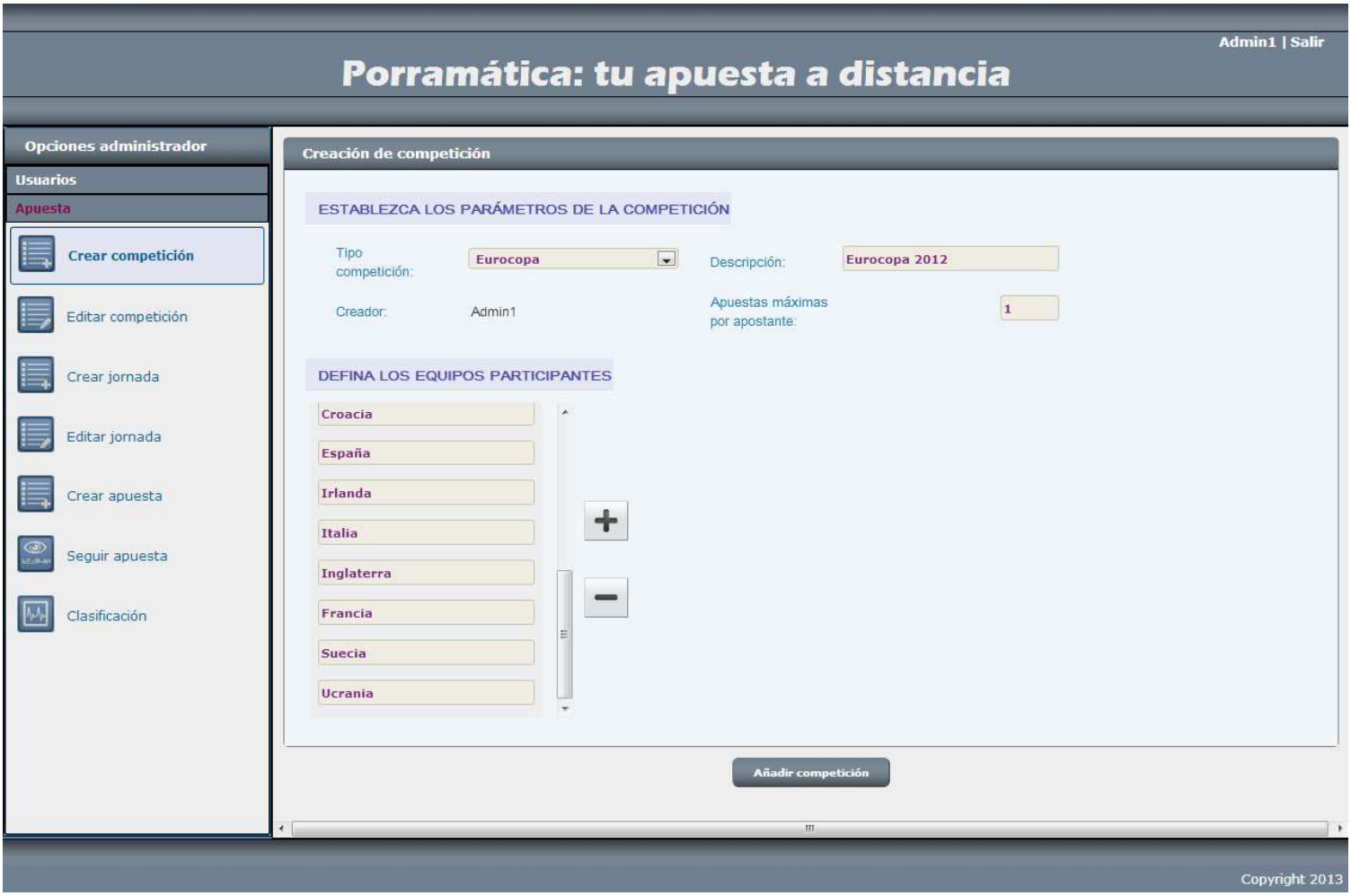

Figura 4.1. Creación competición

El formulario es muy simple e intuitivo. Introducimos el tipo de competición a crear, una descripción, el número máximo de apuestas por usuario por jornada y los equipos que participarán en ella. Aceptamos con el botón “Añadir competición” y ya estará dada de alta en el sistema.

4.2.3 Creación jornadas

Una vez creada la competición, *Porramática* ya acepta la creación de jornadas sobre ella. Por consiguiente el administrador ya puede dar de alta la primera jornada, que en este caso corresponde con el grupo A.

En la siguiente figura podemos ver dicha jornada:

Figura 4.2. Creación jornada

El primer campo que vemos es la competición a la cuál irá asignada la jornada. Seguidamente vemos el tipo de jornada a crear, en este caso es de tipo “Grupo y final” ya que es una jornada donde los usuarios deben apostar tanto por orden de los equipos en el grupo como por el equipo ganador de la competición, por ser la primera jornada en juego.

A continuación vemos la descripción que tenemos que asignar a la jornada, y la fecha máxima de apuesta, que como explicamos en el módulo funcional, sirve para establecer una fecha máxima para apostar en esta jornada.

Procedemos a establecer los partidos que se jugarán en la jornada, y las puntuaciones por cada acción de la jornada, en este caso por acertar signo quinielístico, por acertar el orden de un equipo en el grupo, por acertar todas las posiciones de los equipos en el grupo, por acertar el resultado (que como explicamos sólo se aplica en caso de acertar el resultado completo y significa el número de goles que ha habido en el encuentro por la puntuación que asignemos a este parámetro) y por último la puntuación por acertar el ganador del torneo.

Si nos damos cuenta en el formulario también aparece el botón “Asignar grupo”. Dicho botón sirve para asignar los equipos que formaran el grupo de la jornada a crear. Tan solo debemos añadir los equipos que formen el grupo, como podemos ver en la siguiente figura:

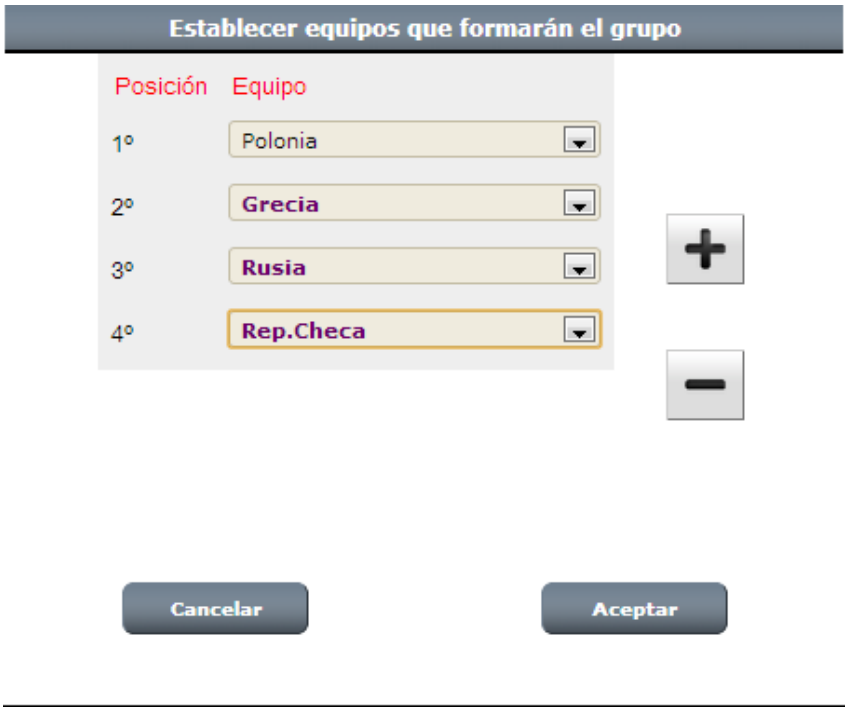

Figura 4.3. Asignación grupo

Una vez asignado el grupo, y si estamos conformes con todo lo establecido, aceptamos la operación mediante el botón “Añadir jornada” y tendremos dado de alta la primera jornada jugable de la competición.

Al finalizar este paso, los usuarios suscritos mediante correo electrónico a notificaciones les llegarán un correo electrónico con los datos de la jornada creada, y de la misma forma veremos en Twitter la información relativa a ella, tal y como figura en la siguiente imagen:

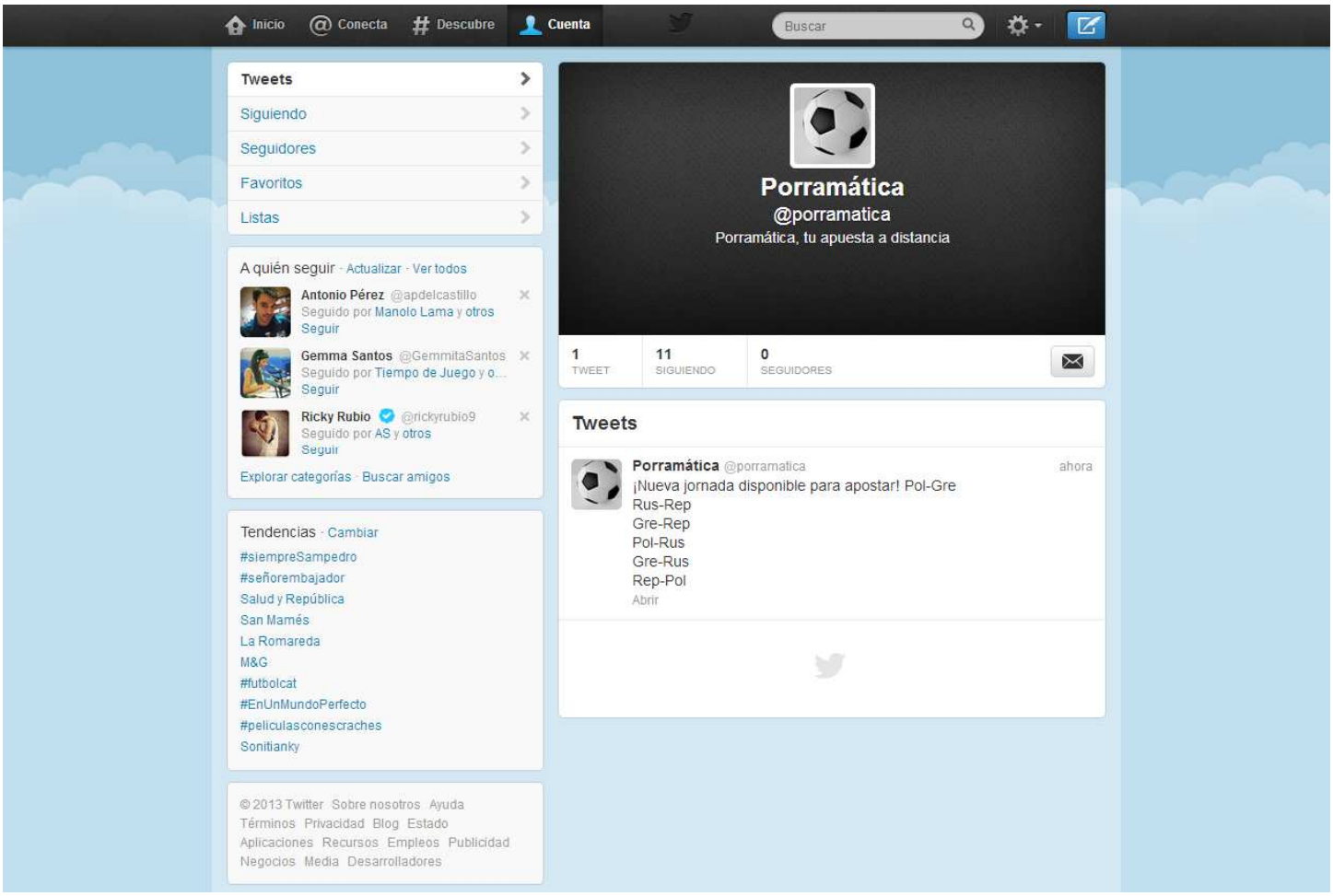

Figura 4.4. Publicación en Twitter nueva jornada

Mostramos también el correo electrónico que ha sido enviado por la aplicación a los usuarios suscriptores:

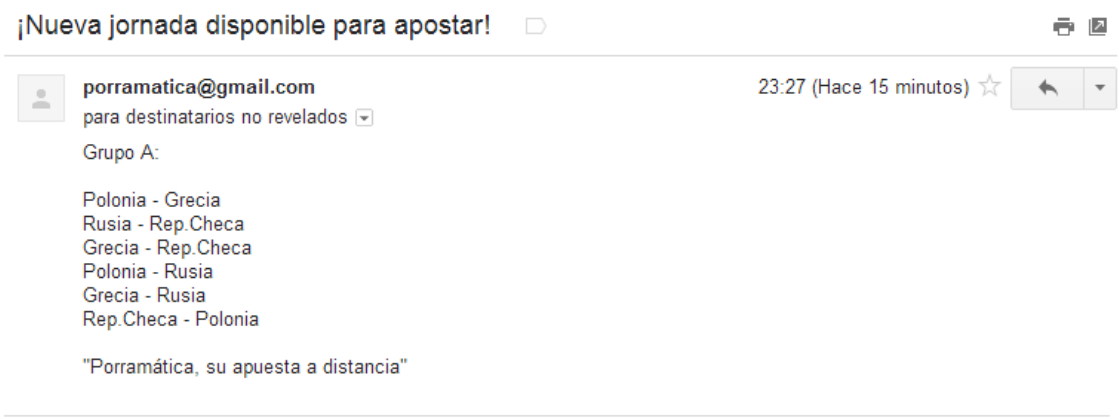

Figura 4.5. Email nueva jornada

4.2.4 Apuesta sobre jornadas

Una vez creada la jornada, es el momento de que los usuarios participantes hagan sus apuestas. Vamos a ilustrar el caso del usuario “User1”.

En su menú “Crear apuesta” se encontrará la siguiente imagen:

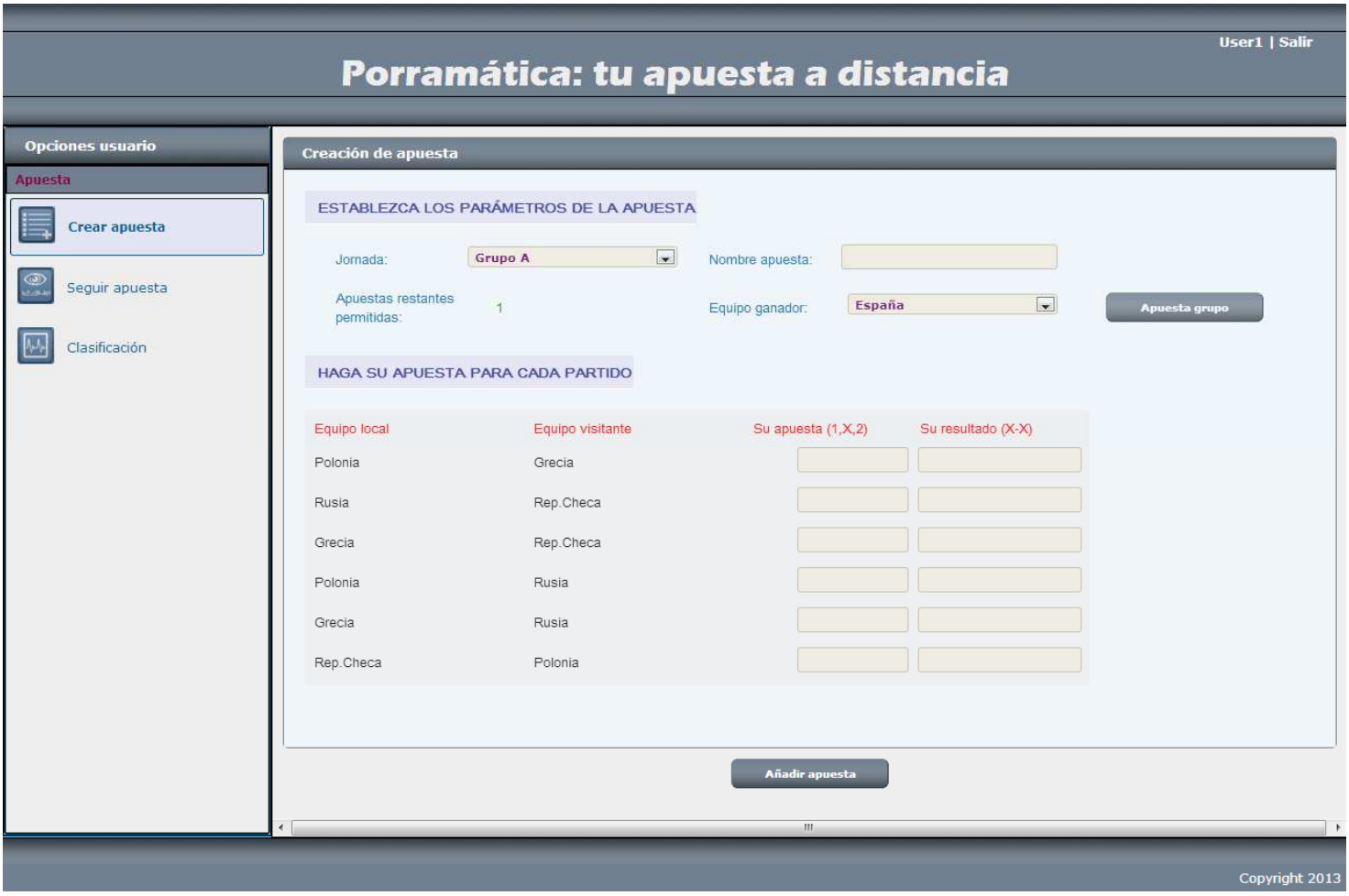

Figura 4.6. Creación de apuesta

El usuario dispone de un menú desplegable donde puede elegir las jornadas activas para apostar, en este caso apostará por el "Grupo A", que es la jornada creada anteriormente.

Deberá asignar un nombre identificativos para su apuesta. También se le recuerda mediante un texto del número de apuestas restantes sobre la jornada seleccionada. Y a continuación los campos característicos de jornada, en este caso una lista con los equipos que forman la competición para que elija el ganador de la competición; botón de "Apuesta grupo" donde deberá elegir el orden de los equipos que juegan la jornada; y la tabla para asignar signo quinielístico y resultado por cada partido a jugar de la jornada.

Una vez relleno todos los datos, pulsará sobre "Añadir apuesta" y su apuesta quedará registrada.

4.2.5 Edición resultados

Si todo marcha correctamente, la fecha máxima de apuesta habrá llegado a su fin y todos los participantes ya habrán realizado sus apuestas. Por lo que el paso siguiente es ir actualizando resultados según el criterio del administrador.

Este paso se realiza en la ventana "Editar jornada" que podemos ver en la siguiente figura:

Figura 4.7. Edición resultados

El usuario dispone de un elemento desplegable para elegir la jornada a editar. Una vez seleccionada le aparecerán las opciones de dicha jornada, en este caso puede seleccionar el equipo ganador del torneo (paso que realizará cuando acabe la competición), puede editar el orden de los equipos en el grupo según vayan sucediéndose los partidos, así como los resultados que se vayan produciendo. También se le permite editar los parámetros de puntuaciones de la jornada.

Una vez editada la jornada según el criterio del administrador, pulsará sobre “Editar” y se hará efectivo el cambio.

A continuación podemos visitar la página de Twitter de la aplicación para encontrar lo siguiente:

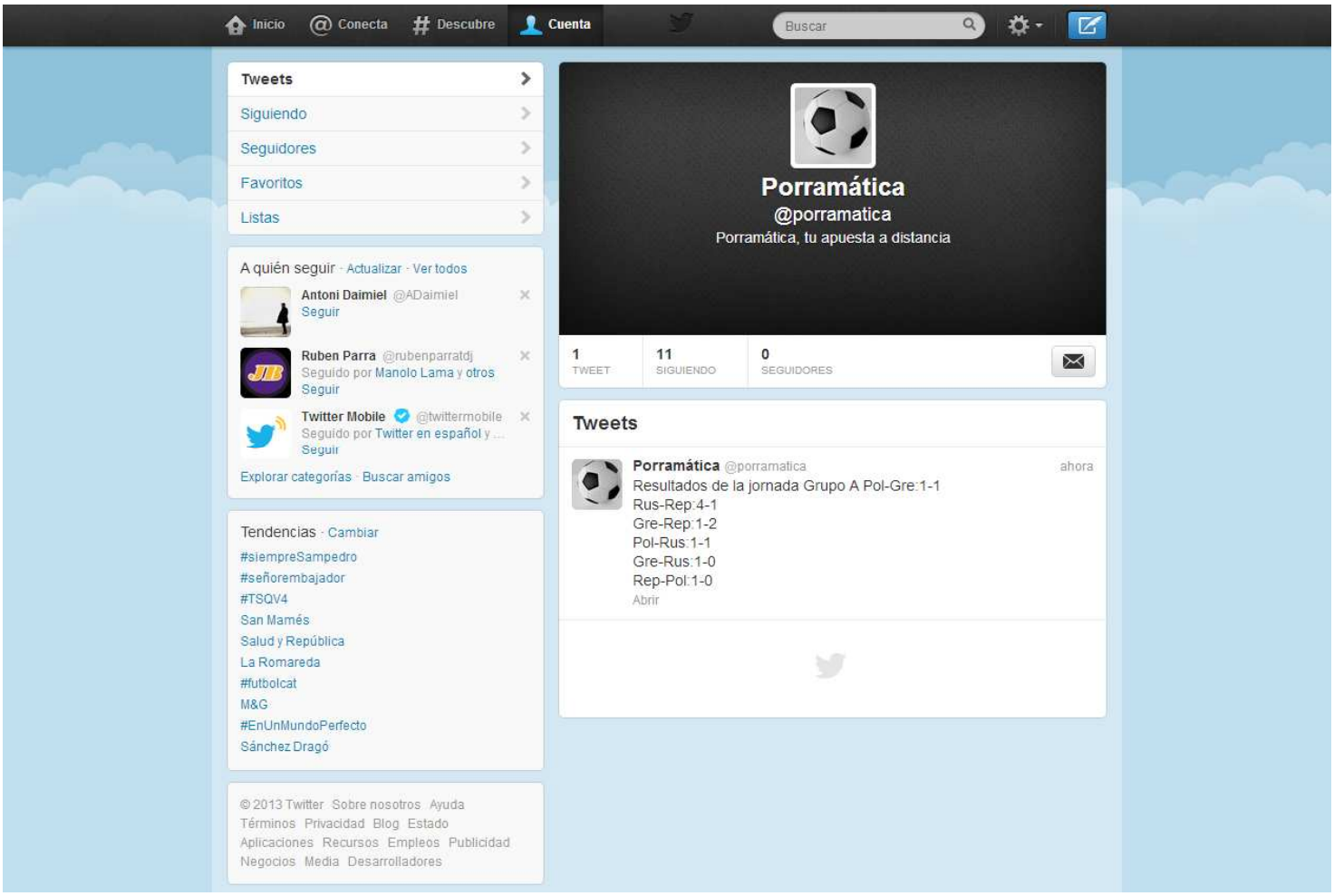

Figura 4.8. Publicación resultados en Twitter

Y de forma parecida, a los usuarios que tengan activada la recepción de alertas por correo electrónico, les llegarán los resultados y se visualizarán de la siguiente forma:

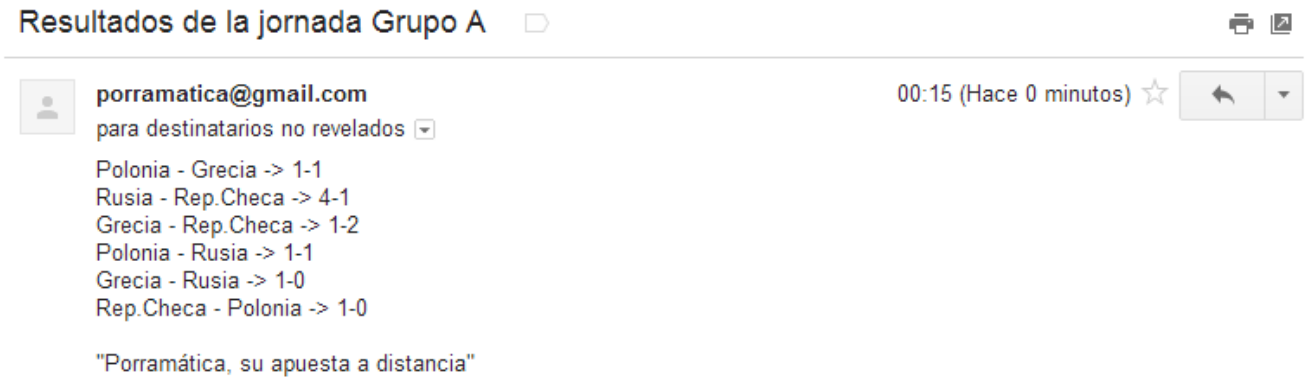

Figura 4.9. Correo electrónico resultados jornada

4.2.6 Seguimiento apuesta y clasificación

Llegados a este punto, el usuario habrá recibido bien vía correo electrónico o informado por la aplicación mediante Twitter de que ha habido cambios en la primera jornada y ya hay resultados de los partidos.

El usuario “User1” primero desea saber cuantos aciertos tiene, por lo que se dirige a la sección “Seguir apuesta” y puede ver la siguiente pantalla:

Figura 4.10. Seguimiento apuesta

El usuario puede seleccionar la jornada que quiere visualizar para ver los últimos resultados actualizados, para posteriormente seleccionar la apuesta que quiere mostrar y corroborarla con los resultados de los partidos.

Para hacer más simple el proceso de recuento de aciertos y puntuaciones, *Porramática* pone a disposición del usuario el botón “Escrutar” que le mostrará la siguiente ventana:

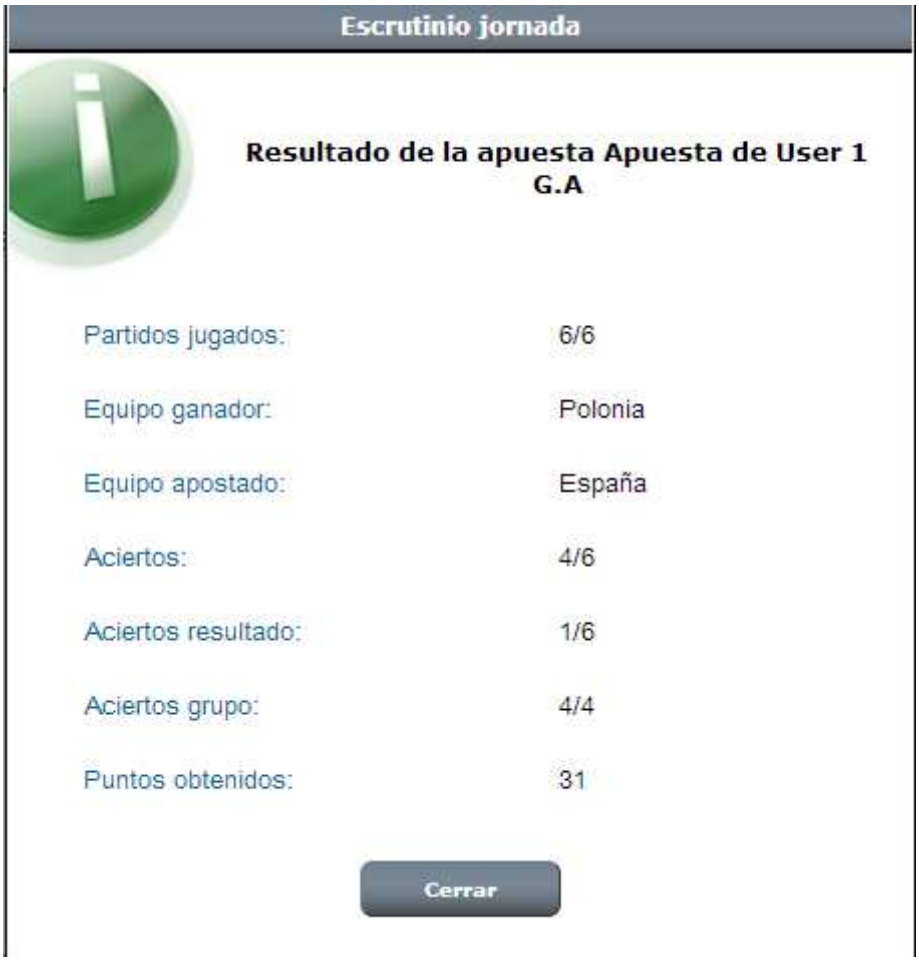

Figura 4.11. Escrutinio apuesta

En ella podemos leer la siguiente información, que variará dependiendo de las opciones de la jornada a visualizar (en este caso vemos todas las opciones posibles al tratarse de una jornada "Grupo y final"):

- Partidos jugados. El número de partidos disputados hasta el momento.
- Equipo ganador. Equipo ganador de la competición. Se editará al finalizar la competición.
- Equipo apostado. Equipo apostado por el usuario para ganar la competición.
- Aciertos. Calcula los aciertos quinielísticos del usuario respecto a los resultados de los partidos.
- Aciertos resultados. Muestra los resultados acertados por el usuario.
- Aciertos grupo. Muestra los aciertos de posiciones de los equipos en el grupo de la jornada.
- Puntos obtenidos. Suma de todas las puntuaciones de la jornada visualizada.

Podemos hacer el cálculo detallado para corroborar que efectivamente *Porramática* ha hecho el cálculo perfectamente:

Puntos obtenidos = (4 aciertos x 3 puntos) + (1 acierto resultado [1-1] = 2 goles x 2 puntos) + (4 aciertos grupo x 3 puntos) + (3 puntos por acertar el grupo entero) = 12 + 4 + 12 + 3 = *31 puntos*.

Por último al usuario le queda por saber cual es su puesto en la clasificación respecto al resto de participantes de la porra. Esto lo puede consultar en la sección "Clasificación" que muestra la siguiente ventana:

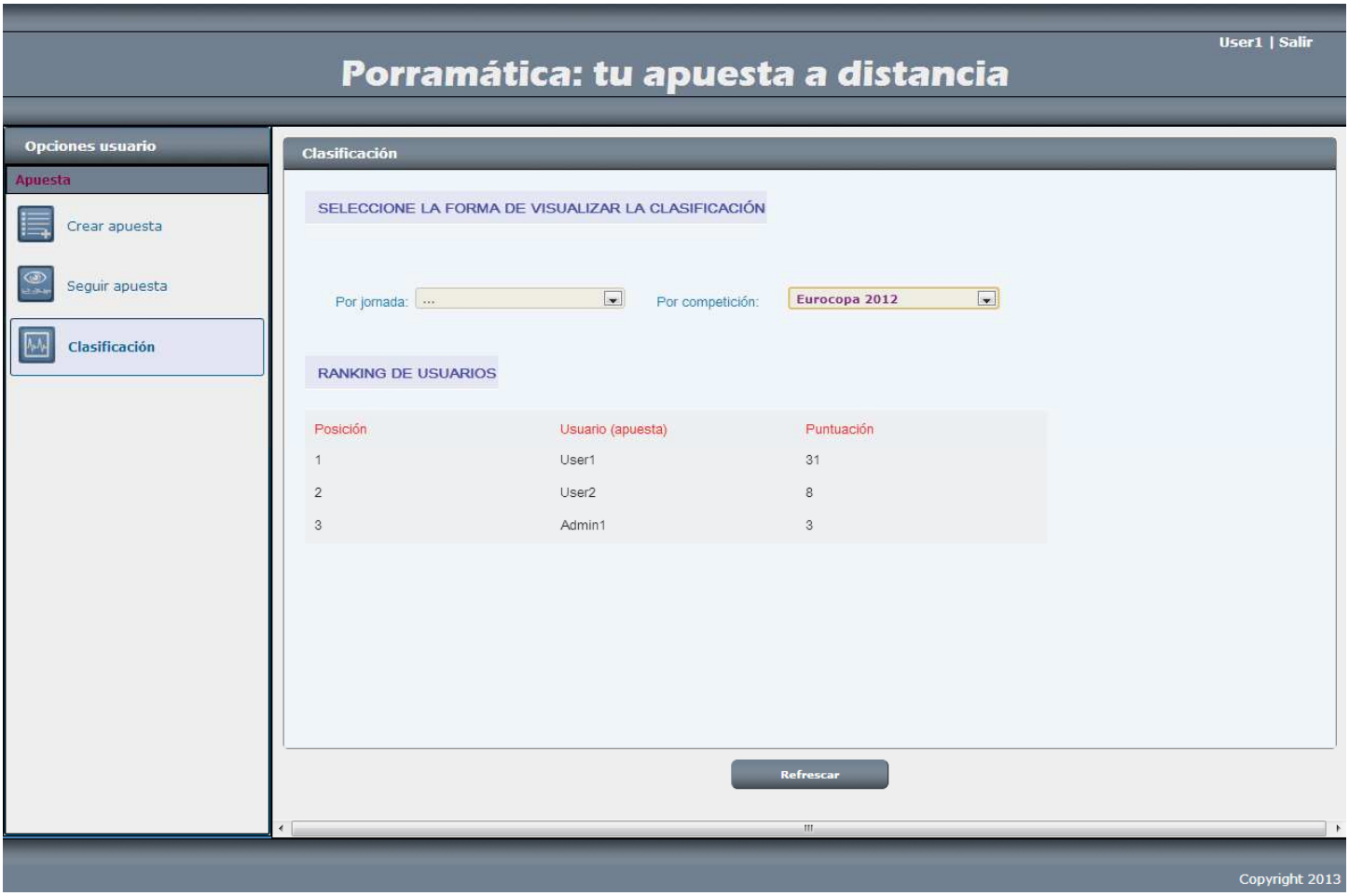

Figura 4.12. Clasificación

En esta ventana el usuario tiene la opción de visualizar la clasificación por jornada disputada o por competición.

En caso de elegir por jornada, se mostrarán las diferentes apuestas realizadas por cada usuario para la jornada en cuestión seguido de la puntuación obtenida en cada una.

Y en caso contrario, si elige la clasificación por competición, *Porramática* elabora la clasificación sumando todas las puntuaciones de las jornadas disputadas de la competición. El ejemplo es el mostrado en la figura B.12.

Capítulo 5

Historia del proyecto

Este proyecto surge como propuesta de realizar una aplicación para gestionar apuestas deportivas por parte de los tutores. Se afronta como un reto por parte del alumno de abarcar tal cantidad de configuración por parte de la aplicación, y ser capaces de dotar de tecnología a la aplicación con el envío de correos electrónicos y tuiteo de información.

Se concreta en desarrollar la aplicación en GWT y el proyecto empieza en junio de 2012.

5.1 Distribución temporal

Este proyecto se ha desarrollado en un tiempo aproximado de 10 meses completos, desde el 1 de junio de 2012 hasta el 31 de Marzo de 2013.

A continuación se describen brevemente las fases del proyecto junto al tiempo dedicado a cada una de ellas. En la figura 5.1 se muestra y se explica el diagrama de Gantt y las fases del proyecto.

Nombre de la tarea	Duración	Comienzo	Fin	jun'12	jul'12	ago'12	sep'12	oct'12	nov'12	dic'12	ene'13	feb'13	mar'13
Realización PFC "Porramática"	**10 meses**	**01-jun-12**	**31-mar-13**										
Documentación	1 mes	01-jun-12	01-jul-12										
Implementación código	4 meses	02-jul-12	30-oct-12										
Diseño interfaz	1 mes	31-oct-12	01-dic-12										
Pruebas	1 mes	03-dic-12	03-ene-13										
Redacción memoria	3 meses	04-ene-13	31-mar-13										

Figura 5.1. Diagrama de Gantt de la realización del proyecto

5.1.1 Documentación

El primer paso de este proyecto es la documentación en la cual se adquieren los conocimientos en los siguientes campos.

- GWT: Conceptos, desarrollo e implementación
- API JavaMail
- API JDBC
- Tuiteo de información en Twitter

5.1.2 Implementación de código

En esta fase, además de la programación como tal del código, se realizó la búsqueda de la información necesaria para la implementación del concepto de la aplicación y la solución de problemas específicos que iban surgiendo: problemas de despliegue en versión final como la comunicación con la parte del servidor, problemas de estilos específicos de la versión de GWT, problemas de plugin el entorno de desarrollo, pruebas no locales con el equipo de la universidad a través de la VPN, etc.

5.1.3 Diseño interfaz

En esta fase se realizó el diseño visual de la aplicación. Debemos mantener una simplicidad y coherencia con este aspecto para que la aplicación sea agradable, accesible y fácil de usar.

5.1.4 Pruebas

En esta fase se realizaron todo tipo de pruebas para comprobar el correcto funcionamiento de la aplicación en los módulos de que dispone la aplicación. Cuando pudimos tener acceso al equipo de la universidad y poder desplegar en el entorno real, nos centramos en los errores que pueden suceder en ejecución, así como los problemas encontrados para desplegar la versión.

5.1.5 Redacción memoria

La realización de la memoria se lleva a cabo en un periodo aproximado de 3 meses íntegros dedicados para ello.

Durante el desarrollo del código y documentación de las tecnologías se redactaron algunas partes de la memoria aunque no está reflejado como tal en el diagrama de Gantt.

5.2 Presupuesto del proyecto

En este apartado se presenta el presupuesto del proyecto, estableciendo tanto el coste material como el del trabajo de las personas que han participado en su desarrollo.

5.2.1 Costes de personal

Los costes de personal incluyen los honorarios del Ingeniero Técnico de Telecomunicación en Telemática encargado del desarrollo del proyecto. La duración de este proyecto ha sido de aproximadamente 10 meses. Podemos aproximar que si estimamos una dedicación de 4 horas diarias continuas y suponiendo 20 días laborables al mes se obtiene un total de 200 días laborables, con una jornada laboral aproximada de cuatro horas diarias, la realización del proyecto ha requerido de 800 horas.

Según el baremo orientativo del Colegio Oficial de Ingenieros Técnicos de Telecomunicación (COITT) en septiembre de 2009 [22] los honorarios de un Ingeniero Técnico de Telecomunicación en Telemática eran de 62 euros la hora, por tanto el coste total ascendería a 49.600€. La siguiente tabla recoge este resultado:

Concepto	Horas	Honorarios	Importe
Ingeniero Téc. Telec. Telemática	800	62€/hora	49.600 €

Tabla 5.1. Costes de personal

5.2.2 Costes de material

Los materiales empleados durante la realización del proyecto han sido los siguientes:

- Un ordenador portátil Acer con sistema operativo Windows Vista de 32 bits valorado en 700€.

- Conexión a Internet durante la realización del proyecto. Ha sido necesaria para conseguir documentación y para probar la aplicación. Está valorada aproximadamente en 43€ al mes [23], que multiplicado por los meses de trabajo, supone un coste de 430 euros.

Teniendo en cuenta todos estos elementos, los costes de material se detallan en la siguiente tabla:

Concepto	Unidades	Precio/ud	Importe
Ordenador	1	700 €	700 €
Conexión ADSL	1	43€/mes	430 €
TOTAL			**1.130 €**

Tabla 5.2. Costes de material

5.2.3 Presupuesto total

El presupuesto final para la realización de este proyecto está formado por los costes de material y de personal presentados anteriormente. Como se observa en la siguiente tabla, el total asciende a 50.730€.

Concepto	Importe
Coste de personal	49.600 €
Coste de material	1.130 €
TOTAL	**50.730 €**

Tabla 5.3. Coste total

Capítulo 6

Conclusiones y trabajos futuros

6.1 Conclusiones

En este proyecto se ha realizado un estudio de la tecnología GWT, APIs de terceros como han sido JavaMail o JDBC, y una aplicación para Twitter que permite el tuiteo de información a través de un correo electrónico.

Empleando tecnologías de software libre, hemos creado una aplicación lo suficientemente potente para ser empleada en un ámbito reducido de usuarios como para ser empleado masivamente y ser capaz de soportar un gran tráfico de información.

En el momento de la redacción de esta memoria no se ha experimentado con ella en un entorno con un número de usuario masivo (mayor de 100), por lo que uno de los retos siguientes es el despliegue de la aplicación para este uso, su escalabilidad y su rendimiento ante esta situación.

6.2 Líneas futuras

Tras el desarrollo de la aplicación *Porramática* y pensando en proyectos futuros que complementen el proyecto desarrollado, a continuación se proponen algunas posibles líneas de trabajo futuro.

6.2.1 Despliegue y uso en entorno masivo

Como hemos comentado en las conclusiones, sería muy interesante enfocar la aplicación para un uso masivo de clientes que se registren, apuesten y sean capaces de valorar el rendimiento de la aplicación.

Si se quiere orientar la aplicación para competir contra rivales reales que cotizan el mercado de las apuestas, véase Bwin, MiApuesta o Quinielista, este debe ser el punto de partida a tomar para seguir desarrollando la aplicación e intentar imitar las fortalezas de los rivales para hacerse un hueco en el mercado.

6.2.2 Funcionalidad nueva: uso de dinero

La característica principal de una casa de apuestas es el movimiento de dinero que se genera por las apuestas. Si queremos continuar con la primera línea futura tratada, es decir, que la aplicación soporte ese flujo de usuarios, lo lógico es que la aplicación adquiera la funcionalidad de poder apostar dinero.

Cada usuario dispondría de un “monedero” en el cuál puede depositar el dinero que él desee. Se puede hacer por tarjeta de crédito o por ingreso bancario. Una vez que el usuario dispone de crédito para jugar, las jornadas dispondrían de un bote, el cual se reparte según las apuestas y la cantidad de dinero que cada usuario haya apostado.

Esto dotaría a la aplicación de una funcionalidad muy seria y de una gran tecnología.

6.2.3 Desarrollo de una aplicación nativa para Android/iOs

Porramática es una aplicación web accesible desde cualquier navegador. Debido al uso masivo de smartphones que hay en el mercado, sería recomendable hacer una aplicación propia del sistema operativo en cuestión que conectará con Porramática y realizara las mismas tareas que se pueden realizar vía web.

Es interesante por dos razones; una aprender el lenguaje de programación que se elija para desarrollarlo (Android o iOS) de manera que el futuro estudiante que lo amplíe pueda desarrollar su carrera en este ámbito, y por otro lado, las aplicaciones nativas hacen un mejor uso de los servicios que ofrecen los smartphones como son la cámara, la libreta de contactos, la ubicación, notificaciones push, etc.

En términos de distribución, las aplicaciones nativas obtienen una buena visibilidad entre los consumidores debido a que son distribuidas a través de las tiendas de aplicaciones del sistema operativo que lleve el teléfono (App Store en el caso de iOS y Play Store, antiguo Android Market, en el caso de Android). Esto puede suponer un modelo de ingresos por las descargas de la aplicación.

Apéndice A

Glosario de términos

AJAX (Asynchronous JavaScript And XML)

API (Application Programming Interface)

CCS (Cascading Style Sheets)

GWT (Google Web Toolkit)

HTML (HyperText Markup Language)

IMAP (Internet Message Access Protocol)

J2EE (Java Enterprise Edition)

JDBC (Java DataBase Connectivity)

JNDI (Java Naming and Directory Interface)

JVM (Java Virtual Machine)

POP3 (Post Office Protocol 3)

RMI (Java Remote Method Invocation)

SGBD (Sistema Gestor Base de Datos)

SMTP (Simple Mail Transfer Protocol)

SQL(Structured Query Language)

Apéndice B

Referencias

[1.] Java 2. Manual de usuario y tutorial, 4ª edición actualizada a la versión J2SE5. Froufe Quintas, Agustín. Ra-Ma Editorial, S.A. 1ª imp. 06/2005.

[2.] Java. Artículo de la Wikipedia. Accesible en http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29

[3.] Artículo online extraído de Scribd, una de las bibliotecas online más grandes del mundo. "Java, su historia, ediciones, versiones y características como plataforma y lenguaje de programación". http://es.scribd.com/doc/19475538/Java-Su-Historia-Ediciones-Versiones-y-Caracteristicas-Como-Plataforma-y-Lenguaje-de-Programacion

[4.] Página web de Oracle, actualmente poseedora de la licencia sobre Java: http://www.oracle.com/technetwork/java/index.html

[5.] Java a tope: JavaMail. Universidad de Málaga. http://www.lcc.uma.es/~galvez/ftp/libros/JavaMail.pdf

[6.] Página oficial de JavaMail. http://www.oracle.com/technetwork/java/javamail/index.html

[7.] Configuración de JavaMail para el envío de emails a través de una cuenta de GMail. http://support.google.com/mail/answer/13287?hl=en

[8.] Página oficial de JDBC http://www.oracle.com/technetwork/java/overview-141217.html

[9.] Página oficial de MySQL para descargar conector JDBC http://dev.mysql.com/downloads/connector/j/

[10.] Enlace Wikipedia de GWT http://es.wikipedia.org/wiki/Google_Web_Toolkit

[11.] Página oficial de GWT https://developers.google.com/web-toolkit/doc/latest/DevGuide

[12.] Serialización y llamadas remotas https://developers.google.com/web-toolkit/doc/latest/DevGuideServerCommunication

[13.] RPC https://developers.google.com/web-toolkit/doc/latest/tutorial/RPC

[14.] Manual referencia MySQL http://dev.mysql.com/doc/refman/5.0/en/

[15.] Enlace Wikipedia SQLYog http://en.wikipedia.org/wiki/SQLyog

[16.] Página oficial de Tomcat http://tomcat.apache.org/

[17.] Enlace Wikipedia Twitter http://es.wikipedia.org/wiki/Twitter

[18.] Enlace a la página de Twittermail http://twittercounter.com/pages/twittermail?ref=footer

[19.] Creación de cuenta en Google para Gmail https://accounts.google.com/newaccount?hl=es

[20.] Creación de cuenta en Twitter https://twitter.com/

[21.] Plugin para Eclipse GWT-Export https://code.google.com/p/gwt-project-export-wizard/

[22.] Proyecto Fin de Carrera "Apuntweets. Herramienta de colaboración sobre materiales educativos" de Carlos Azaustre Rodríguez.

[23.] Tarifas de conexión a Internet de ONO. http://www.ono.es/resources/files/tarifas/pdfs/residencial_actual.pdf